# Technická univerzita v Liberci

Fakulta mechatroniky, informatiky a mezioborových studií


---

Studijní program: B2646 – Informační technologie

Studijní obor: 1802R007 – Informační technologie

# Návrh ontologie pro webové služby B2B

## Design of an Ontology for B2B Web Services

Bakalářská práce


Autor: **Jan Sikorjak**

Vedoucí práce: Ing. Július Štuller, CSc.


V Liberci 17.5.2012

# Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum:

Podpis:

# Poděkování

Děkuji tímto vedoucímu práce, Ing. Júliu Štullerovi, CSc. za cenné zkušenosti, které se mi během tvorby práce snažil předat a také pro jeho tendenci práci zdokonalovat k dosažení odborné formy splňující požadavky přibližující se vědecké práci. Dále bych chtěl poděkovat prof. Dr. Ing. Jiřímu Maryškovi, CSc. za podnět k volbě tohoto tématu, občasné konzultace a doporučení vedoucího práce, který je uznávaným odborníkem v oblasti. Zároveň můj dík patří i Ing. Petru Viktorovi, řediteli IT divize společnosti Deloitte ČR za ochotu k diskuzi v oblasti integrace podnikových aplikací u větších společností na českém trhu. V neposlední řadě děkuji svému kolegovi, Bc. Václavu Burešovi za praktické podněty k obsahu práce a shovívavost v časově náročnějším řešení společných projektů.

## Abstrakt

V první části práce popisuji aktuální trendy v oblasti webových služeb, sémantického webu a souvisejících technologií. V praktické části této BP jsem na jejich základě navrhnul ontologii online B2B služeb a následně konceptuální model databáze skladu pro velkoobchodní či distribuční společnosti. Součástí práce je i specifikace a implementace rozhraní pro online přístup ke skladovým zásobám a základním procesům obchodní korespondence.



## Klíčová slova

Webové služby, Sémantický web, Ontologie, B2B, API



## Abstract

In the first part of the thesis the actual trends in the area of web services, semantic web and related technologies are described. Based on these I designed an ontology of online B2B services followed by a conceptual database model of a storehouse for wholesale or distribution company. The thesis also includes an interface specification and implementation for online access to the supplies and basic business communication processes.



## Keywords

Web Services, Semantic web, Ontology, B2B, API

# Obsah

# Seznam obrázků a tabulek

# 1 Úvod

Rozmach internetu v posledních letech přináší vedle nezpochybnitelných výhod i nutnost adaptace softwaru pro toto prostředí či změnu způsobu ukládání a reprezentace informací. Z pohledu světových statistik mělo začátkem roku 2012 přístup k internetu přibližně 33% populace. Ve srovnání s rokem 2000 to značí nárůst o 528%. Dnešní trendy však poukazují na ještě vyšší tempo, zejména v rozvíjejících se a lidnatých zemích jako Brazílie, Rusko, Indie nebo Čína (souhrnně označovaných také jako BRIC). S ohledem na počet obyvatel těchto velkých zemí a jejich ekonomický růst lze odhadnout, že časem zde může penetrace připojení dosáhnout procentuálních hodnot západních zemí, kde se aktuálně pohybuje okolo 70%. Předpoklad může podpořit fakt, že např. Čína disponuje v absolutních číslech dvojnásobkem uživatelů internetu oproti USA, a to při pouze poloviční procentuální penetraci. Potenciál dalšího růstu je tedy obrovský a již nyní naráží konvenční způsoby pro výměnu, uspořádání a vyhledávání informací na internetu na své limity.

Největší světový internetový vyhledávač Google odhadem zpracovává indexované záznamy z cca 40 miliard webových stránek a elektronických dokumentů. V tomto obrovském množství informací je velice složité se orientovat i pro sofistikované algoritmy na bázi aktuálně využívaných slovníků. Jedním problémem je zařazení dokumentu do správného kontextu, dalším pak třídění výsledků dle relevance a v neposlední řadě i rozpoznání kvalitních zdrojů. Důkazem toho může být fakt, že Google v roce 2011 oficiálně uvedl 16 balíčků se změnami ve vyhledávacím enginu a obecně v posledních letech vyvíjí stále větší aktivitu z hlediska inovací v hodnotících algoritmech.

Pokud tedy vezmeme v potaz odhady pro další růst počtu připojených uživatelů k internetu a s tím očekávatelné větší množství informací na internetu sdílených, dojdeme k závěru, že jednou z cest, jak se vyhnout mnohem většímu informačnímu přehlcení, může být jistá restrukturalizace dat ve smyslu přiřazení strojově zpracovatelného významu k jejich částem tak, aby nejen vyhledávače, ale i speciální prohlížeče a agenti mohli poskytnuté dokumenty automaticky procházet a řadit. Jinými slovy definicemi významů převést data na informace z perspektivy počítačového zpracování. Těmto otázkám se věnuje oblast tzv. sémantického webu a část práce pojednává právě o této problematice z hlediska používaných technologií a dnes již dostupných a používaných nástrojů.

Vedle sémantického webu také dochází ke znatelným změnám v architektuře aplikací a informačních systémů. Stále větší počet aplikací se tak postupně přesouvá na internet do

formy webových aplikací a populárním se stává i cloud[1], pomocí kterého lze provozovat i celou virtualizovanou serverovou platformu pro běh více aplikací či celých systémů, které mají být přístupné přes internet. Nutno také zmínit boom v oblasti tzv. „chytrých mobilních zařízení“ a dalších typů klientů, které lze k internetu připojit (např. televize, tablety, domovní bezpečnostní systémy apod.).

Internet umožňuje propojit množství nejrůznějších aplikací či zařízení, a aby spolu mohly efektivně komunikovat, bylo potřeba uvést nové technologie, které budou v ideálním případě využívat již zavedené síťové protokoly a budou tak pro autory aplikací univerzálně, jednoduše a rychle použitelné. Vznikly tak webové služby – veřejná a standardizovaná rozhraní, prostřednictvím kterých mohou spolu dvě aplikace vyměňovat data a zprávy přes internet (resp. jakoukoliv počítačovou sít) nezávisle na programovacím jazyce či běhovém prostředí.

Praktickou součástí práce je spojení obou výše uvedených technologií (sémantický web a webové služby) a vytvoření modelu pro použití sémantických webových služeb v obchodní korespondenci organizace zapojené v dodavatelsko-odběratelském řetězci. Vedle definice business procesů a jejich zachycení v ontologii je výsledkem i databáze pro uchovávání informací o nabízených produktech s důrazem na maximální míru abstrakce a katalogizace. V konečné fázi je pak možné pomocí online rozhraní z databáze číst a naopak do ní zapisovat s ohledem na metody a omezení vymodelované přiloženou ontologií. S využitím webových služeb a jejich snadné integrace v systémech klientů tak lze urychlit a zefektivnit proces např. dotazování na skladové množství a ceny, následně nákup, expedici atp. Při akceptaci jedné společné ontologie nebo namapování ontologie jednoho subjektu na ontologii partnera tak bude v budoucnu možno tyto procesy zcela automatizovat.

[1] Cloud – pronájem výpočetní i diskové kapacity a provoz aplikací na serverech poskytovatele služby umístěných do velkých datacenter, často geograficky odlišených se snahou o maximální škálovatelnost a abstrakci hw.

# 2 Webové služby

Webové služby, anglicky „Web Services“, slouží jako standardizovaný prostředek pro integraci aplikací přes počítačovou síť. Technologii lze uvažovat jako aplikační rozhraní při návrhu architektury softwaru a řeší i komunikační stránku věci. Většinou jsou tyto služby provozovány přímo na webovém serveru vedle běžných internetových stránek, protože sdílí shodný komunikační model i protokol (standardně se s nimi pracuje přes HTTP protokol s požadavky GET a POST, příp. požadavky typu REST, které uvedu později). Na rozdíl od běžných stránek však webové služby fungují jako knihovna funkcí, které přímo poskytuje hostitelská aplikace. Požadavek se zpracovává na serveru a klientovi je předán výsledek v předem uvedeném formátu (nebo pouze stavová informace, pokud se tato služba využila způsobem tzv. vzdáleného volání procedury).

Pro zatím obecnou představu z hlediska architektury si uveďme, že se jedná o kombinaci jazyka WSDL pro standardizovaný popis komunikačních metod a funkcí, které aplikace poskytující webové služby nabízí, dále jazyka XML pro specifikaci a značkování předávaných dat, protokolu SOAP pro výměnu dat jako takovou (SOAP umožňuje komunikaci nejen prostřednictvím protokolu HTTP, ale také SMTP, popř. jiných) a adresářové služby UDDI pro vyhledávání dostupných služeb od různých vydavatelů a provozovatelů.

Implementace webových služeb do aplikací je dnes velice snadná; vývojáři takřka nemusí rozlišovat mezi tvorbou běžných bloků programového kódu a částí obsahujícím webové služby. Obecně je totiž poskytnuta vysoká míra abstrakce a pohodlí, kdy se téměř vytrácí prvek komunikace samotné, a tak se vývojáři mohou soustředit zejména na vlastní logiku aplikace, což činí vývoj efektivnějším. Díky rozvoji internetu a dříve naznačené potřebě transformace aplikací jsou webové služby důležitým prvkem ve vývoji online světa. Namísto leckdy náročných implementací spojení typu klient/server přes přímé TCP/UDP socketové připojení je možno v ideálním případě nabízenou službu přímo lokalizovat prostřednictvím jejího URL, automaticky namapovat na objekty používané v klientské aplikaci a pouze již volat dostupné metody, jakoby se jednalo o obyčejné funkce v programovém kódu lokální aplikace. Funkce webových služeb v principu vychází z logiky vzdálených volání procedur (RPC[2]), byť jejich dnešní klasické použití se ubírá spíše směrem SOA (Service Oriented Architecture - architektura zaměřená na službu a předávání určité zprávy a dat s očekáváním odpovědi namísto jednoduchého spuštění procedury na serveru), kdy je možné prostřednictvím nich provádět

[2] RPC, Remote Procedure Call – technologie pro spouštění procedur na jiných počítačích.

transakce se strukturovanými daty, dotazovat se serveru a výsledky ve formě typovaných dat zpětně přijímat.

V důsledku je pak možno služby skládat a kombinovat. Takový kompozit se nazývá *mashup* a s ohledem na aktuální zvyklosti se hovoří spíše o uživatelských prvcích na webových stránkách (např. mapy Google zobrazující umístění a recenze k restauracím získané od jiného zdroje). Řetězit či kombinovat lze buď přímo na straně klienta, nebo nějakým dalším serverem v pozici prostředníka (brokera). Jednou z oblastí, kde lze skládání služeb aplikovat, je obchod typu B2B, kde technologie může napomoci např. automatizaci procesů (objednávky, logistika, účtování, …), o které budeme hovořit později v praktické části práce a která dnes naráží především na absenci uznávaných standardů.

## 2.1 WS jako součást architektury IS

Připomeňme, že z hlediska architektury informačních systémů můžeme rozlišovat aplikace dle počtu a typu používaných vrstev. Obecně může v aplikaci figurovat N vrstev, kde každá z nich má nějakou specifickou funkci (prezentační, datová, logická atp.). Dělení do takových bloků je často způsobeno požadavkem na abstrakci nebo výkon a jeho škálovatelnost (tj. v případě nutnosti rozšíření počtu zařízení, na kterých aplikace či její část běží). Musíme tak věnovat zvláštní pozornost jednotlivým rozhraním mezi vrstvami, které se, jak bylo uvedeno výše, mohou vyskytovat v různých počítačích (např. střediska logistických společností mohou být rozesety po celém světě). Programy nebo i celé systémy mohou na takových místech poskytovat pro cizí či externí aplikace rozhraní ve formě tzv. API (Application Programming Interface). Nemusí se přitom jednat pouze o hranice jednotlivých vrstev; obecně API slouží pro vnější přístup k aplikaci z jiných programů nebo softwarových komponent. S příchodem webových služeb odpadá nutnost využití konvenčního API; není tak potřeba využívat cizí knihovny nebo nižší síťové protokoly. Naopak – webové služby díky jednotnému standardu nabízí vyšší míru abstrakce a nezávislosti na prostředí nebo programovacím jazyku. Jistou nevýhodou API řešeného webovými službami je samozřejmě větší režijní náročnost na systémové prostředky a výkon z důvodu zapouzdření do jiných technologií. Nespornou výhodou použití webových služeb je také průchodnost požadavků a odpovědí přes firewally, jelikož zde využívají již většinou povolených nosných protokolů, respektive jejich přiřazených sítových portů.

## 2.2 Popisovací jazyk WSDL

Pro standardní popis webových služeb z hlediska funkcionality se využívá Web Service Definition Language (WSDL) na bázi jazyka XML. Samotný WSDL je tedy z formálního hlediska

spíše schématem složeným z XML elementů popisujících jednotlivé služby, funkce a jejich datové typy. Obecně se ale hovoří o popisovacím jazyku (ostatně je to uvedeno i v názvu této specifikace). První verzi (WSDL 1.0) uvedly v roce 2000 společnosti Microsoft a IBM jako výsledek kompromisu mezi svými projekty. Aktuální verze 2.0 je již pod dohledem W3C[3] konsorcia a rozdíl mezi těmito verzemi spočívá právě v upřednostnění SOA architektury namísto původního využití typu RPC.

WSDL se skládá z několika základních stavebních bloků – typy (**types**) definující strukturu dat pomocí XSD schémat pro požadavky i odpovědi jednotlivých funkcí služby, dále pak rozhraní (**interface**) pro přehled nabízených funkcí, vazby (**bindings**) propojující rozhraní s povolenými typy, resp. formáty spojení (budou popsány dále) a konečně služby (**services**), které popisují cesty k funkcím služeb s ohledem na definované vazby.

## 2.3 Protokol SOAP

Sada pravidel pro komunikaci webových služeb, Simple Access Object Protocol (SOAP), je postavena na jazyku XML a přímo vychází z původního konceptu XML-RPC, odkud využívá nezávislost na přenosovém médiu a dalších výhod použitého značkovacího jazyka. Protokol je však závislý na ostatních protokolech Aplikační vrstvy ISO/OSI modelu (např. již uvedené HTTP či SMTP), je to tedy protokol vyšší úrovně (analogie s programovacími jazyky nižší a vyšší úrovně). Původní XML-RPC navíc SOAP obohacuje o rozdělení zprávy do struktury obálka-hlavička-tělo známé z paketů jiných síťových protokolů. Akronym SOAP může naznačovat i jistou podobnost se zkratkou SOA – byť spolu v důsledku souvisí, jejich názvy nenesou žádnou přímou vazbu či spojitost.

**Síťový přenos a volání**

Nejčastější předání dat mezi webovou službou a jejím klientem (označovaným také jako konzument) se uskutečňuje v těle POST HTTP požadavku. Alternativou, která však již nefunguje na bázi XML, a tím pádem vylučuje využití SOAP, může být i předání nestrukturovaných dat s primitivními datovými typy v podobě parametrů při požadavku typu GET volaném spolu s URL webové služby – to je vhodné například pro ladění nebo také vyhledávání dle jednoduchých neobjektových klíčů a parametrů (číslo, řetězec…) nebo pro použití služby jako RPC. Služba však vždy ve formátu SOAP odpoví, pokud je specifikován ve WDSL.

[3] W3C, World Wide Web Consortium – organizace spravující webové strandardy

Existují i způsoby, jak do zprávy zakomponovat přílohu typu MIME pro zaslání např. obsahu souboru nebo binárních dat. Nejpoužívanější metoda pro zasílání příloh se nazývá SwA (SOAP with Attachments).

**Výhody**

Jednoznačná výhoda protokolu SOAP tkví ve strukturovanosti a detailním popisu formátu předávaných dat, což umožňuje snazší vývoj, ladění a nezávislou činnost s ohledem na platformu nebo klientský a serverový programovací jazyk, ve kterém jsou aplikace vzájemně komunikující přes webové služby navrženy či sestaveny.

Další výhodou je do určité míry závislost na běžně používaných protokolech, a tedy odpadá nutnost definice dalších síťových protokolů. Tzv. tunelování přes tyto protokoly pak umožňuje již zmíněný průchod firewally, což možná do budoucna bude pro tyto protokoly představovat problém z hlediska zabezpečení, jelikož trend využívání nosných protokolů se pravděpodobně nezastaví a přes HTTP(S) se bude poskytovat stále více dalších služeb a nadstaveb, takže standardní pojetí paketového firewallu bude poněkud postrádat smysl.

**Nevýhody**

Nevýhody SOAP vesměs dědí od svého stavebního kamene – XML: nutnost specifikace použitých schémat pro validaci dokumentu spolu s uváděním několika jmenných prostorů (namespaces) pro všechny tagy. Ty jsou navíc následkem své přirozené parity uváděny v dokumentech alespoň 2x, obsahují-li další vnořené elementy. Následkem předešlých faktů tak narůstá objem předávaných dat – ovšem problém se netýká pouze přenosu, ale také náročnosti procesu sestavení a zpětného parsování celého XML stromu. Každopádně toto není záležitost většiny běžných použití, kde se přenáší méně rozsáhlé datové struktury a jde spíše o časté a jednoduché požadavky, obzvláště pak u případů využití webových služeb způsobem RPC. Nabízí se ale alternativa jednoduchého zápisu předávaných dat pomocí JSON[4].

Druhou a relativně zásadní nevýhodou je absence jakékoliv přímo zakomponované autorizace, šifrování či komprese. V tomto se SOAP čistě spoléhá na schopnosti nižších protokolů, proto zatím jediná použitelná metoda pro zabezpečenou komunikaci je HTTP+SSL (Secure Socket Layers), resp. kompresi typu GZIP opět definované prostřednictvím HTTP hlaviček. Samozřejmě ale je možné zavést vlastní proprietární řešení pro tyto účely.

[4] JSON, JavaScript Object Notation – technologie pro serializaci dat

## 2.4 Požadavky typu REST

V současné době se nabízí i varianta transferu dat přes REST (Representational State Transfer) rozhraní, což je doplněk ke známým HTTP požadavkům a nabízí oproti tradičnímu procedurálnímu RPC (a tím pádem i jeho následníkovi SOAP) spíše zaměření na jednotlivé entity a práci s nimi. REST, někdy také pro svou jednoduchost nazývaný jako „RESTful services", tak zavádí nové požadavky a upravuje funkci požadavků již zavedených: GET, PUT, POST, DELETE a snaží se o zpřístupnění funkčního modelu CERD[5], někdy také označovaného jako CRUD[6] pro plnohodnotnou správu dat přes webové služby. Pro přenos dat je jednodušší – nevyžaduje přítomnost XML. Namísto toho strukturuje a serializuje data pomocí JSON. Nevýhodou je ale horší čitelnost při vývoji a ladění, kdy na první pohled není zřejmé pořadí jednotlivých hodnot a jejich hierarchické vnoření.

Ostatní vlastnosti jsou již shodné se SOAP, jelikož pramení z HTTP protokolu. Ještě pro úplnost doplňme zásadní nevýhodu HTTP protokolu – absence možnosti udržování stavově-perzistentního připojení, kdy neexistuje dlouhodobá komunikační relace mezi serverem a klientem a po každém požadavku, resp. odpovědi na něj dojde k odpojení.

## 2.5 Vývoj webových služeb

Nástroje pro tvorbu webových služeb jsou dnes velice sofistikované a bývají běžnou součástí vývojových platforem. Zároveň si nutno uvědomit, že webové služby jsou doménou spíše větších aplikací nebo systémů, které je typicky potřeba propojit s jinými systémy. Tomu odpovídá i míra integrace u vývojových nástrojů či používaných programovacích jazyků. Nyní si uvedeme krátký přehled možností práce s webovými službami, které nabízí dvě nejpoužívanější platformy pro webové aplikace – jazyk PHP a architektura ASP.NET.

**PHP**

U dnes nejrozšířenějšího webového skriptovacího jazyka PHP je tvorba webových služeb poměrně nepohodlná. Nejenže v základní sadě funkcí chybí možnost dle datového modelu automaticky generovat WSDL, a tím se nadále nestarat o tuto definici při případných následných úpravách, ale také chybí možnosti pro řízení systémových objektů a prostředků (např. větvit provádění kódu do tzv. vláken pro výkonovou optimalizaci a paralelizaci programu na procesorech s více jádry nebo multiprocesorových architekturách; obecně operace s vlákny nejsou z hlediska vlastností PHP možné).

[5] CERD – Create, Edit, Read, Delete

[6] CRUD – Create, Read, Update, Delete

Z pozice konzumenta či klienta služeb již lze celkem jednoduše vytvořit ve zdrojovém kódu objekt z WSDL schématu a následně s ním pak pracovat jako s běžným objektem, tj. volat přímo procedurálně dostupné metody, které vrací výsledky. Avšak pro tyto účely je potřeba zavést do PHP rozšíření pro SOAP nebo jinou technologii. Buď ve formě systémové knihovny SoapClient a jejího povolení v konfiguračním souboru, nebo použitím externích knihoven (např. populární wsdl2php nebo NuSOAP).

Obecně je tedy PHP určeno s ohledem na dostupné nástroje a možnosti spíše jako klientská strana a tvorba webových služeb je zde náročnější. V mnoha případech bude potřeba sáhnout po komponentách třetích stran, což PHP v tomto smyslu znevýhodňuje.

**ASP.NET**

Framework .NET, použitý v této práci, naopak vývojáři poskytuje možnosti a automatizované nástroje nejen pro tvorbu webových služeb, ale také pro jejich následný provoz, ladění atp. Vývojové prostředí Visual Studio disponuje i projektovými šablonami pro jejich tvorbu. De facto vše, co je pak potřeba provést během vývoje takové služby, je napsat serverový obslužný kód ve vybraném jazyce (C#, VB.NET…) - sem lze implementovat na rozdíl od PHP i řízení toku kódu do již zmíněných vláken nebo vytvářet tzv. session spojení (které dokáže na serveru uchovat stavové informace klienta např. o přihlášení, což sám protokol HTTP neumožňuje).

Další výhodou je automatické vygenerování vizuálního webového klientského rozhraní, takže pokud do svého internetového prohlížeče zadáme URL adresu webové služby, zobrazí se formuláře pro volání funkce s možností přímého zadání vstupních hodnot prostřednictvím textových polí. WSDL dokument můžeme jednoduše zobrazit tak, že zadáme **[URL služby]?wsdl** a získáme tak automaticky vygenerovanou definici opět sestavenou ze zdrojového kódu (a která je navíc uložena v serverové cache z výkonnostních důvodů).

Jako poslední nadstandardní funkci ještě uvedu objektové mapování z pozice klienta. Typicky požadujeme do své aplikace zakomponovat cizí webovou službu. Není pak nic jednoduššího, než pomocí nástroje ve Visual Studiu zadat její URL adresu a přidat ji do projektu. Tím se vygeneruje speciální mezivrstva obsahující metody a dokonce i datové typy použité v připojované službě. Ve svém zdrojovém kódu posléze voláme z vytvořeného objektu příslušné metody již běžnou syntaxí a nemusíme tak odlišovat volání lokálních metod od volání metod webových služeb.

## 2.6 Provoz webových služeb

Jak už bylo zmíněno v úvodní kapitole k webovým službám, běžně se hostují na serverech jako součásti webových aplikací. Zpravidla tak není potřeba pro jejich provoz do systému instalovat dodatečné komponenty či obslužný software. Maximálně v některých případech je bude nutno explicitně povolit (např. v IIS[7] 6.0 jsou defaultně vypnuty). Víme také, že jsou jednoznačně identifikovány svojí URL adresou a fungují na protokolu HTTP. Z tohoto hlediska se tedy od běžných webových skriptů reprezentujících stránky nikterak neliší a není potřeba jejich provoz dále odlišovat. Zásadní rozdíl spočívá v jen tom, že handler (obslužný kód, který operaci řídí) přesměruje požadavek pro webovou službu na patřičný modul, který již komunikaci zajistí a abstrahuje ji klientovi pro další vykonávání.

Pro ad hoc účely je samozřejmě možno vyvinout i vlastní serverovou aplikaci; jen je nutné odlišit port, na kterém bude „naslouchat“, aby nekolidoval se standardním webovým serverem – tzn. pravděpodobně zvolit port s jiným číslem, než standardní 80. S tímto způsobem se setkáváme třeba u informačních systémů typu ERP[8], CRM[9] a dalších, které nemusí přímo využívat webový server a komunikaci z integritních či bezpečnostních důvodů obstarávají samy.

## 2.7 Registr UDDI

Aby společnosti využívající a poskytující webové služby mohly vzájemně najít cestu, kterak spolu prostřednictvím nich komunikovat a aby při navazování obchodních kontaktů nebylo potřeba vyměňovat dokumentace ke službám, vznikl v roce 2000 veřejný adresář webových služeb UDDI – Universal Description Discovery and Integration. Ten sestává ze třech základních registrů: **White Pages** pro evidenci institucí poskytujících webové služby, **Yellow Pages** jako katalogizace služeb na základě amerických standardů SIC[10] a NAICS[11] a v neposlední řadě **Green Pages** informující o používaných protokolech a dalších technických aspektech jednotl. služeb.

Zásadním zlomem v rozvíjejícím se provozu UDDI bylo v roce 2006 odstoupení společností Microsoft, IBM a SAP z firemního registru. Popularita a využití UDDI s postupem

[7] IIS – Internet Information Services, aplikační a webový server určený pro platformu MS Windows

[8] ERP – Enterprise Resource Planning, systém pro řízení zdrojů a procesů v podniku

[9] CRM – Customer Relationship Management, systém pro řízení vztahů se zákazníky

[10] SIC – Standard Industrial Classification, standard pro klasifikaci odvětví průmyslu a služeb

[11] NAICS – North American Industry Classification Systém, podobný standard jako SIC

času klesá a dnes lze hovořit o úpadku, jelikož jsou tyto adresáře málo flexibilní vůči rychlému tempu vývoje webových služeb a sémantického webu.

Aktuálně však neexistuje náhrada v podobně ekvivalentní a dostatečně respektované služby a je otázkou, kdy se na trhu objeví nějaká nová technologie, která dokáže služby lépe vyhledávat a bude přijata i velkými společnostmi a odbornou komunitou.

## 2.8 Zabezpečení

Zabezpečit webové služby na úrovni přenosu je možné použitím kombinace SSL a HTTP protokolu, známé jako HTTPS. Další možnosti je zašifrovat přenášený XML soubor technologií „XML Encryption“ standardizovanou W3C konsorciem, avšak byla již v minulosti prolomena a není mezi odbornou veřejností příliš oblíbena. Jelikož použití webových služeb se týká většinou obchodních či výrobních společností, kdy jsou přenášena citlivá data, je potřeba navíc obsah zprávy a zejména identitu autora ověřit digitálním podpisem. Problém řeší XML Signature (ve zkratce XML Dsig), taktéž vedený pod záštitou W3C.

# 3 Sémantický web

Potřebu budování sémantického webu vyjádřil a řešení navrhnul „otec internetu“, Tim Berners Lee již v roce 2001. Jedná se o jakýsi cíl transformace, kdy budou dnešní a budoucí webové zdroje doplněny strojově zpracovatelnými informacemi o významu a vazbách tak, aby je mohly automatizované nástroje a zařízení správně vyhledat, zpracovat a reprezentovat. Bohužel technologie v dnešní době zatím nejsou schopny samy správně určit významy poskytnutých informací v dokumentech a musíme tak zavést způsoby a metodiky pro manuální či poloautomatický (tzv. supervised) zápis těchto významů a vazeb. Sémantický web se odkazuje na společenskovědní obor - sémantiku, která je zkoumána již od dob antických a zabývá se právě vztahy znaků (slov) a skutečnostmi.

## 3.1 Ontologie

Důležitým pojmem v oblasti sémantického webu je ontologie. Podobně jako u sémantiky hovoříme o původně filozofické disciplíně. Základní ideou ontologie v informatice je zachycení znalosti v dané doméně (problematice). Cílem je sestavování znalostních bází ve formě slovníků a schémat, které napomáhají k dalšímu zkoumání daného jevu či oblasti, dále pak v důležité automatizaci procesů nebo pro informační účely (generování dokumentace, schémat, nápovědy…). Znalostí se zde v pravém smyslu slova značí jakákoliv popsatelná problematika – např. předmět, proces nebo i celý systém, kde figurují jak předměty, tak procesy. Podobně jako k uchovávání dat slouží databáze, ontologie slouží jako nástroj pro reprezentaci znalostí v počítači.

**Stavební prvky ontologie**

Základem pro budování ontologie je **jedinec**. Pod jedincem si představme jakýkoliv objekt (člověk, strom, automobil, zeměkoule…) či abstraktní pojem (datum, číslo, věta, aktivita, proces…).

Skupina jedinců určitého typu, ať už myšlenkového nebo závislého na společných atributech, se nazývá **třída** a ta může obsahovat další třídy (**podtřídy**). Pomocí nich tak vzniká hierarchický systém třídění a klasifikací jedinců (taxonomie).

**Atributem** nazveme typizovanou vlastnost vztaženou k jedinci nebo třídě. Součástí atributu musí být název, resp. určený typ (např. hmotnost, barva…) a jeho hodnota, která může být reprezentována číslem, řetězcem anebo i jiným objektem v ontologii. Jedince lze dostatečně popsat množinou těchto atributů. Navíc díky dělení jedinců do tříd můžeme ve spojitosti s atributy hledat nové souvislosti a na jejich základě odvozovat nové třídy pomocí

speciálního nástroje, tzv. reasoneru, který však primárně slouží pro automatickou kontrolu chyb v ontologii.

Posledním důležitým nástrojem pro sestavování ontologie je relace (**vazba**), s níž se definují vztahy mezi jedinci či třídami. V závislosti na výběru deskriptivní logiky lze vztahy reprezentovat různými způsoby – jedním z nich může být orientovaný graf, kde třídy a jedinci jsou vrcholy a hrany zastupují právě ony vztahy. Díky tomu lze specifikovat jednosměrné i recipročí vztahy z reálného světa. K vazbám je možno ještě přiřadit několik druhů funkčních atributů (např. určení jednosměrného vztahu, omezení rozsahu hodnot apod.)

## 3.2 Popisovací a značkovací jazyky

**RDF**

Základním značkovacím jazykem určeným pro popis webových zdrojů a vycházejícím ze struktury XML je RDF (Resource Description Framework). Jeho podstatou je definice tzv. trojic (triples), jednoduchých spojení s následující šablonou: **podmět (subject) – predikát – předmět (object)**. Tyto trojice mohou vyjádřit jakoukoliv potřebnou závislost a např. tak anotovat data k nějakému dokumentu. Vedle obvyklého způsobu uložení této jednoduché struktury v relační databázi existují i tzv. *triplestores*, proprietární databáze optimalizované pro efektivní skladování a dotazování na velké množství trojic.

Vraťme se nyní k jednotlivým částem trojice: podmět v oblasti webu může reprezentovat popisovaný dokument, resp. odkazuje na konkrétní URL/URI a k němu je posléze přiřazen predikát, označující typ vztahu (často vyjádřen slovesem) s hodnotou uloženou v předmětu. Příkladem tak může být výrok „*http://www.abc.cz - autor - Jiří Novák*“.

Zajímavostí jazyka RDF je možnost popsání výroku jiným výrokem, kupříkladu, že „*Jiří Novák tvrdí, že autorem dané stránky je René Hužva“*. Máme tedy možnosti pro řetězení a vnořování výrazů, kde se předmětem stává jiný výrok. Jazyk RDF je však nedostačující pro reprezentaci kompletního modelu ontologie, jelikož postrádá definice tříd a v současnosti se využívá spíše pro jednodušší značkování dat např. v elektronických vizitkách (vcard), RSS zdrojích (kde RSS je dokonce akronymem slov RDF Site Summary) a jiných vesměs exportních formátech.

**RDF Schema**

Aby bylo možno formálně lépe popsat ontologii, byl jazyk RDF přepracován do nové podoby přidáním hierarchického systému tříd a vlastností, možnostmi omezení oborů hodnot pro jednotlivé třídy, vytyčení hranic domén atp. Stále však na původním RDF staví ve smyslu reprezentace dat – tvorbou schématu se plní slovník, resp. databáze trojic. Informace uložené

v RDFS (RDF Schema) je možné z databáze dotazovat speciálním jazykem SPARQL, který má podobnou syntaxi jako původní SQL, avšak je doplněn o funkce určené k dotazováním speciálně nad RDF daty.

Nově můžeme ve schématu vznést třeba následující výrok: „*Savci jsou podskupinou Zvířat*“. Všimněme si, že zde na rozdíl od RDF definujeme novou třídu, která je zároveň podtřídou jiné třídy.

**Rodina jazyků OWL**

Jazyky typu OWL (Web Ontology Language) jsou považovány díky své komplexnosti za nejvhodnější znalostní jazyky pro sémantický web. Vlastnosti dědí od svého předchůdce, konkrétně jazyka DAML+OIL, nicméně nalezneme odlišnosti jako např. možné definice symetrických vlastností v OWL, rozdíly v názvech používaných prvků, tzv. primitiv (OWL využívá především notaci z RDFS) a další. Navíc třídám poskytuje funkce pro kombinace, separace a dále pak umožňuje třídy abstraktně definovat či porovnávat.

Výhoda v možnosti zapsání i složitých modelů z reálného světa s sebou nese na druhé straně nevýhodu ve smyslu větších nároků na zpracování (např. při hledání souvislostí, odvozování nebo výpočtech). Vedle plnohodnotného OWL FULL, který je zpětně kompatibilní s RDF, avšak zároveň svou expresivitou zvyšující výpočetní náročnost či dokonce nevyčíslitelnost některých závislostí, existují i následující deriváty s určitými omezeními v sadě dostupných definicí:

- OWL DL – plně vystačuje pro tvorbu ontologie, je ještě kompatibilní s RDF
- OWL Lite – verze s omezenými možnostmi pro zachycení znalostí, jednoduchý ve vyjadřování (na druhou stranu je zde reprezentace komplexních vztahů pracnější) a obecně již nekompatibilní s RDF

## 3.3 Sémantické webové služby

Nyní máme k dispozici dostatečný technologický základ pro návrh a vývoj webových ontologií a jejich aplikaci na webu. Lze tedy postoupit dále a začít uvažovat nad integrací sémantiky do webových služeb s cílem maximální možné automatizace celé řady procesů, které v současnosti musí vykonávat sami uživatelé nebo se na nich nějakou mírou podílí. Zdroj (1) uvádí, že právě tato oblast bude stěžejním prvkem budoucí 3. průmyslové revoluce.

Pro další výzkum je nutno si stanovit základní požadavky na takovou automatizaci a diskutovat jejich možnosti plnění s ohledem na aktuální stav a vývoj oblasti webových služeb a sémantického webu:

1. **Automatické vyhledávání webových služeb**
2. **Automatické volání (invokace) webových služeb**
3. **Automatické skládání (composition) webových služeb**
4. **Automatické monitorování probíhajících procesů pomocí webových služeb**

První bod, automatické vyhledávání, je v současnosti jedním z nejdiskutovanějších problémů. Jedinou dosavadní možností pro dosažení této funkčnosti je využití veřejného adresáře webových služeb UDDI. Bohužel diverzita kategorizace a metod identifikace služeb u různých poskytovatelů těchto registrů znemožňuje automatické vyhledávání na bázi typů a parametrů. Snaha o obejití a vyhledávání služeb v registrech pomocí klíčových slov taktéž může končit neúspěchem. Ostatně nemáme garanci úspěchu ani při vyhledávání ručním způsobem, natož pak automaticky.

I přesto, že bychom požadovanou službu dokázali pomocí UDDI najít, nemáme stále k dispozici sémantické informace a nemůžeme tak ani ověřit, zda výsledná služba je ta, kterou skutečně hledáme, případně zda jsou předávaná data z hlediska významu přesně těmi, která potřebujeme zpracovat. Nicméně UDDI poskytuje přístup k WDSL definicím služeb, takže pokud by existovala možnost, jak do WDSL doplnit sémantické značkování, vyřešila by se tím velká část nastíněného problému s vyhledáváním a ověřováním správnosti výsledků.

Diskuze ostatních bodů již tematicky vybočuje nad rámec práce, bude tedy vynechána.

**Rozšíření WDSL-S**

Výhodou integrace sémantických informací do WDSL definic je jednoduchost a znovupoužití dostupných standardů. Podobně jako u značkování HTML tagů lze značkovat i definiční soubor webové služby za účelem snazšího vyhledání a příp. identifikace příslušných funkcí či typů použitých v odkazované ontologii (např. pro nalezení a párování procesů mezi různými ontologiemi). V patřičných XML elementech tak stačí doplnit odkaz na externí ontologický model popisovaného objektu či problému a k jednotlivým operacím definovat jejich ekvivalenty v onom modelu.

Spolu s možnostmi odkazování na ontologické zdroje a jejich mapování nabízí WDSL-S také sémantickou anotaci podmínek pro vstupní data, výsledných efektů a zařazení služby do kategorií registrů UDDI. Některé UDDI katalogy již tedy nabízí možnost mapování sémantických atributů k vlastním typům a kategoriím. Obecně slouží rozšíření WDSL-S k řešení vyhledávacích problémů a není jeho snahou popisovat další aspekty související s oblastmi automatizovaného zpracování. K tomuto účelu slouží složitější a sofistikovanější nástroj, OWL-S.

**OWL-S**

Pro obecný popis účelu, činností a zařazení webových služeb je OWL-S standardem. Vedle anotace předávaných dat a poskytovaných funkcí umožňuje OWL-S popsat navenek i službu jako takovou například právě pro účely vyhledávání, automatického volání, skládání s jinými službami a monitorování – jinými slovy při správné implementaci ontologií na různých úrovních lze napomoci řešení celkové automatizace komunikačních procesů, jejíž problémy byly nastíněny výše.

OWL-S bývá také označována jako „vyšší ontologie“, jelikož se skládá ze tří typů tzv. *subontologií*, kde každá z nich pak definuje určitý obor znalostí o službě a jako celek dostatečně službu popisuje ze všech potřebných ohledů:

- **Profilová ontologie**, nazývaná také jako „Service Profile“, popisuje webovou službu navenek, řadí jí do určitého kontextu a slouží k účelům propagace a vyhledávání. Vedle strojově čitelných údajů je důležitá i čitelnost těchto informací pro uživatele. Zároveň obsahuje informace o vydavateli, funkčních omezeních atp. Nezasahuje však do znalostní domény celkové modelované ontologie, kterou webová služba řeší.
- **Procesní ontologie**, resp. „Process model“ slouží již k popisu funkčních částí služby. Nalezneme zde definice typů vstupních dat, logických hodnot a výstupních výsledků.
- **Grounding ontology** definuje, jak je služba technicky dostupná. Součástí této ontologie je popis podporovaných protokolů, formátů zpráv a dalších informací technického charakteru.

Díky rozsáhlému zdokumentování je možno se odpoutat od omezených možností registrů UDDI a dát tak vzniku vyhledávacích robotů, které mohou procházet jednotlivé služby a v závislosti na poskytnutých sémantických informacích je indexovat a třídit.

# 4 Cíl práce a současný stav v B2B prostředí

Z mých osobních zkušeností soudím, že většina malých a středních obchodních společností má problém s formou evidence informací k jimi nabízeným produktům stejně jako se špatně nastavenými komunikačními procesy – ať už s využitím internetu nebo bez něj.

Všimnul jsem si i případu, kde výrobce určité komodity postrádá jakýkoliv katalog svých výrobků a odběratelé jsou tedy odkázáni na zdlouhavou komunikaci s obchodními zástupci. Podobný problém se vyskytuje např. u zahraničního distributora herních titulů, který sice využívá ERP systém SAP a odběratelé mají k dispozici pravidelně aktualizované informace v excelové tabulce zasílané e-mailem, ale tyto sestávají pouze z názvu, unikátního interního kódu produktu, ceny a počtu kusů skladem. Dotazování např. na dostupné jazyky u hry nebo její minimální hardwarové nároky si člověk musí zajistit dohledáváním z jiných internetových zdrojů. V oblasti, kde se obchoduje s tisíci různými položkami, to pak obnáší mnoho zbytečných starostí navíc. Snahou výrobce či dodavatele by přitom mělo být poskytnutí co možná nejdetailnějších informací k produktům vč. popisků, typizovaných atributů pro vyhledávání a porovnávání, správného zařazení do stromu kategorií (taxonomie), obrázků, manuálů a dalších souborů.

Dalším častým problémem je absence jakékoliv snahy o automatizaci obchodních procesů, která je dnes výsadou snad jen těch největších organizací. Rozesílání ceníků e-mailem, následné ruční zadávání objednávek a celá jejich správa vč. jednotlivých stavů od zaslání *pro forma* faktury až po dodání sledovacího kódu odběrateli je dnes i u výhradních dovozců vysokoobrátkového zboží běžně zajišťována tzv. account managery a spočívá tedy v oboustranném osobním kontaktu. Připomeňme, že na této bázi jsou obě strany omezeny pracovní dobou, objemem práce, který je schopen člověk v určitém čase vykonat a také větší pravděpodobností výskytu lidských chyb. Je to možná cílem firemní politiky, kdy se organizace chce orientovat na budování osobních vztahů s odběrateli, ale tato metoda nekoresponduje s možnostmi současných technologií a rozchází se s trendem automatizace těchto procesů, která bude v blízké budoucnosti takřka nutností pro konkurenceschopnost a vůbec fungování takové organizace na trhu.

Řešení těchto nedostatků se nabízí formou implementace tzv. ERP systému, jehož moduly umí mj. skladovou evidenci i obchodní korespondenci přes internet zajistit. Bohužel jsou dnešní ERP systémy většinou nadstavbou účetních programů rozšířených o další moduly nebo jsou naopak až zbytečně složité a pro mnoho firem nepoužitelné či drahé. Možnosti pro evidenci informací o sortimentu jsou tak omezené nebo jsou výsadou těch nejdražších verzí. Zároveň dnešní podnikové systémy pochopitelně nereflektují trendy a moderní postupy, a to

především ze dvou důvodů: nedůvěra v budoucnost mladých technologií (nejsou standardizovány a akceptovány většinou) a případná potřeba častých úprav a revizí, kdy by se de facto každý měsíc mohly provádět aktualizace s novými funkcemi. Malá flexibilita pro změny je jedním z největších úskalí rozvoje automatizace procesů v této oblasti. Ostatně podobně tomu bylo i u dalších technologií, které se musely nejdříve vyvinout, standardizovat a zaplatit v komerční či vojenské sféře, aby byly posléze masově uvolněny mezi veřejnost (např. GPS, ale i internet samotný).

Součástí této práce a jedním z cílů je tedy identifikace běžných obchodních procesů, resp. služeb, které by se mohly stát základem pro vývoj nového B2B obchodního systému či modulu do ERP systému, zaměřeného na integraci systémů partnerů a která by tak napomohla nejen k automatizaci např. objednávek, ale také k lepší distribuci informací o produktech. Dnes je běžnou praxí, že menší a střední organizace využívají několik různých systémů pro různé účely namísto jednotného a komplexního řešení v podobě drahého ERP. Právě pro tyto případy, jako doplněk do stávajícího portfolia systémů, může být řešení navržené v této práci odpovídající.

Omezení, s kterými se však takové řešení potýká, je nedostupnost uznávaných standardů, případně nedostatečná obsáhlost a akceptace již zavedených a spíše „místních“ standardů pro klasifikaci produktů či obchodních procesů. Dalším bodem, který vyplynul ze spíše teoretického charakteru práce, je formální nedokazatelnost správnosti výsledku, jelikož se jedná o adaptaci nových technologií do prostředí vymezeného zavedenou praxí a výsledek tedy nelze jednoznačně ohodnotit; bude zde hrát roli osobní pohled a zkušenosti s problematikou. Ontologie lze ohodnotit pouze nasazením v praxi a následně zdokonalovat iterativním procesem. Otázkou stále zůstává použitelnost konečného návrhu ontologie, kdy teoreticky neexistuje ideální řešení pro danou problematiku. Jistou alternativou pro účely práce však může poskytnout optimalizace a měření výkonu databáze, která byla sestavena tak, aby byla schopna uchovat data k prvkům obsaženým v přiložené ontologii.

# 5 Návrh řešení – datový a komunikační model

Prostředky popisované v první části práce, jmenovitě webové služby a ontologie, se vzájemně doplňují a výsledný efekt lze využít v mnoha aplikacích. Výsledkem práce je návrh principu komunikace obchodních systémů různých subjektů. Pro tento účel označím webové služby jako ideální technologii, která je nenáročná na integraci, běžně používaná a při šikovné implementaci dokáže nahradit osobní komunikaci doposud zajišťovanou zaměstnanci. Otázku takové implementace, která má být v ideálním případě automatická a vylučující případné nesrovnalosti v pochopení významů propojených procesů i předávaných dat obou subjektů řeší vytvoření ontologie. Avšak není nutné ihned zavádět sémantické webové služby; ontologie zachycující vztahy použitých procesů a dat může stejně tak vhodně posloužit jako dokumentace či stavební prvek databáze a vůbec celého systému, který bude obchodní komunikaci zajišťovat. Proto jsem obě technologie využil v praktické části. V následující kapitole pro úplnost ještě uvedu dnešní nejpoužívanější metody pro integraci systémů, kterým mají sémantické webové služby konkurovat.

## 5.1 Současné integrační metody

Snaha o integraci podnikových aplikací (EAI[12]) na bázi zasílání zpráv přes síť či internet je známa již od 70. let. Tehdy OSN uvedla jednotný standard pro obchodní komunikaci – **EDIFACT**[13]. Dnes je hojně rozšířen v USA a Evropě, kde byla EAI tímto způsobem zavedena ještě před příchodem novějších metod. EDIFACT zároveň definuje vlastní komunikační protokol a syntaxi, což jej činí složitějším a dražším pro nasazení. S příchodem jazyka XML ale vznikly další metody – **ebXML** a **RosettaNet**, které jsou flexibilnější a intuitivní i bez náhledu do dokumentace. Používanější z nich, RosettaNet dokonce definuje vlastní standardy klasifikace produktů a některé parametry pro B2B procesy.

Nicméně všechny uvedené metody naráží na možnosti automatizace, jelikož jsou zde jednotlivá data i metody předávány buď bez strojově zpracovatelného významu, nebo obsahují pouze omezené sady standardizovaných parametrů a stále je potřeba „ruční“ integrace na obou stranách vč. vedení dokumentací. Univerzální a jednoznačný popis metod i předávaných dat lze vymodelovat pomocí nějaké deskriptivní logiky, reprezentované např. ontologií. Na rozdíl od předešlých způsobů je v tomto případě možno namapovat jednotlivé metody webových služeb přímo na ekvivalentní procesy v ontologii (Process Model). Pro řešení v této

[12] EAI – Enterprise Application Integration (integrace podnikových aplikací)

[13] United Nations/Electronic Data Interchange For Administration, Commerce and Transport, ISO 9375

práci byla tedy zvolena cesta tvorby ontologie a ukázka komunikace prostřednictvím webových služeb a dalších běžně používaných formátů.

## 5.2 Definice stěžejních B2B služeb

Abych mohl přistoupit k tvorbě jednotlivých tříd a vztahů v ontologii, musel jsem nejdříve definovat procesy, které se běžně v obchodní korespondenci vyskytují. Nejedná se o kompletní výčet, ale pouze o základní sadu těch nejdůležitějších služeb, kterých se bude B2B komunikace týkat. V komerčním nasazení takového systému by vznikla potřeba tvorby dalších služeb, jejich dělení do tříd a podtříd, zařazení do procesního workflow apod.

Konzumenti níže uvedených služeb budou vždy ověřeni pomocí jejich uživatelských přístupových údajů, aby byli k takové činnosti autorizováni a systém jim mohl odeslat personalizovaný výstup i v závislosti na nastaveném jazyce a měně. Stěžejním prvkem personalizace výstupu bude však možnost stanovení rozdílných cen pro různé zákaznické skupiny – D1, D2, D3 apod.

**Export produktového katalogu**

První službou v pořadí je export nabízeného sortimentu s uvedením maximálního množství informací o produktech. To obnáší samozřejmě určité nároky na architekturu databáze, aby klient měl možnost data zpětně reprezentovat bez ztráty původních informací. Katalog bude obsahovat výčet všech evidovaných položek vč. parametrů a jejich hodnot, zařazení do stromu kategorií, textové popisky a odkazy na přidružené soubory. Aby se uspokojila globální poptávka, bude možné jako parametr požadavku na export použít identifikátor jazyka, ve kterém budou informace sestaveny a odeslány. Pokud nebude parametr dodán, výstup proběhne ve výchozím jazyce registrovaném u uživatelského účtu, kterým se bude klient autorizovat. Je důležité také poznamenat, že tento výstup nebude obsahovat informace o cenách a skladové dostupnosti, ale pouze statická katalogová data, takže se nabízí možnost uložení vygenerovaného výstupu této služby do serverové cache a snížení výpočetních nároků na hostitelský systém. Aktualizaci katalogu u klientů by bylo ideální provádět jednou denně, případně v závislosti na exportu skladových zásob dle potřeby (např. přibude-li v ceníku nový produkt, který ještě nemá klient neimportován v databázi).

**Export skladových zásob**

Jelikož ke změnám cen a zejména pak skladových dostupností dochází v kratších časových intervalech (u větších společností to může být i několik sekund), je žádoucí, aby export aktuálních stavů byl minimální a neobsahoval zbytečná další data, která se tak často

nemění. Proto došlo k oddělení obchodních dat z produktového katalogu a vytvoření této služby. Podobně jako u exportu katalogu je uživatel identifikován a v závislosti na nastavení profilu bude výstup obsahovat ceny v příslušné měně; opět je možné doplnit do požadavku pro volání služby vlastní volbu měny a přepsat tím jednorázově uložené nastavení.

Jednotlivé ceny lze přiřadit různým zákaznickým skupinám, takže např. odběratel s obratem vyšším než X bude mít výhodnější cenové relace, než odběratel s obratem nižším. Společnosti, které provozují několik skladů či prodejen, mohou v navrženém systému navíc evidovat dostupnosti jednotlivých produktů (resp. variant) pro každý tento sklad zvlášť. Proto budou v exportu skladových zásob uvedeny i dostupnosti (počet jednotek fyzicky skladem) ve všech skladech, které jsou pro něj použitelné, aby klient mohl využít odběru v nejbližším místě v případě, že dodavatel nevyužívá zásilkovou službu či poštu pro distribuci produktů.

Tato diskutovaná služba je významově shodná s běžným ceníkem (v obchodní terminologii také označovaným jako „stocklist“), avšak má tu výhodu, že je k dispozici online a využívá vždy aktuální hodnoty. Doporučuji aktualizaci provádět každou hodinu pro co nejrychlejší zachycení změn.

**Zadání objednávky**

Pro dodání požadovaného zboží je potřeba započít proces vložením objednávky do obchodního systému dodavatele – zadání tzv. **Purchase Order** (PO). Jak již bylo uvedeno výše, v drtivé většině případů u menších a středních společností se tento postup vykonává manuálně a právě tato služba dává celému procesu význam z hlediska možné automatizace. Z hlediska funkce se jedná spíše o požadavek typu RPC, kdy se na serveru spustí procedura zadání objednávky – jako návratová hodnota ale může posloužit stavový kód, že služba byla vykonána. Ve složitějších nasazeních je pro úplnou automatizaci potřeba zavést sofistikovanější odpověď, a to ve formě potvrzení objednávky (**order acknowledgement**), které bude obsahovat seznam položek, které byly zadáním rezervovány a přidány k objednávce. Systém odběratele by takovou odpověď zpracoval, požádal obsluhu o vyřešení (storno vyprodaných položek, příp. umístění do budoucích rezervací, tzv. **backorder** apod.) a s výsledkem opět kontaktoval dodavatelský systém, který by zadání vyhodnotil a potvrdil, případně by zaslal opět seznamy rezervovaných položek (a tento proces bude iterovat, dokud se nevyeliminují problematické položky). Pro účely této práce se spokojíme s prostým potvrzením objednávky, protože implementace vrstvy zajišťující tyto logické operace by přesahovala rozsah práce..

Jako vstupní parametry pro službu bude nutné uvést vedle ověřujících údajů uživatele i seznam objednaných položek (ID nabízené položky + počet objednávaných jednotek), dále

způsob platby a dopravy a případně poznámku pro operátora. Při přijetí v systému dodavatele dojde ke zpracování a uložení objednávky do databáze v částečně denormalizovaném formátu (bude diskutováno později).

**Dotaz na stav objednávky**

Protože objednávka není jednorázová akce, nýbrž proces probíhající několika stavy, které zajišťují různí operátoři, je zásadní, aby systém mohl odběratele o jednotlivých přechodech informovat nebo aby umožnil odběrateli se na stavy dotazovat. Pro účely dotazování pak slouží tato služba. Jako parametr přijímá unikátní identifikátor objednávky, který bude na straně serveru dodavatele ověřen oproti dodaným autentifikačním údajům a jako návratová hodnota poslouží struktura obsahující typizovaný stav, datum a čas změny, označení operátora, který objednávku do stavu přenesl a případně poznámku k tomuto přechodu.

Díky tomu bude moci klientský systém pravidelně kontrolovat dodávky produktů a sám označovat dostupnosti k produktům pro své odběratele či přímo koncové zákazníky (např. zboží je na cestě od dodavatele apod.).

## 5.3 Požadavky pro evidenci produktových informací

Zásadním prvkem pro použitelnost systému z hlediska evidence produktů je dostatečná abstrakce tak, aby výsledná databáze mohla uložit a zpracovat data o produktech bez jejich vynechání či znehodnocení oproti dostupným znalostem z reálného světa. Proto se přiložená ontologie z větší části zaměřuje na specifikaci domény evidence produktů a dalších přidružených objektů.

**Společné vlastnosti produktů**

Hlavním elementem znalosti, kterou chceme reprezentovat, je produkt a zde si definujeme společné vlastnosti, které musí všechny evidované a zpracovávané produkty obecně obsahovat. Z důvodu již uvedené potřeby abstrakce se k objektu budou vztahovat pouze následující atributy:

- Jednoznačný identifikátor (ID)
- Název (v různých jazycích)
- Textový popis (v různých jazycích)
- Měrná jednotka (definovaná v systému pro znovupoužití, tzv. reuse)

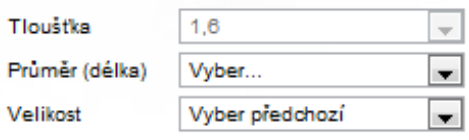
**Parametry**

Abychom mohli plně využít dynamickou klasifikaci produktů např. pro vyhledávání, filtrování, generování personalizovaných nabídek nebo následně vyhodnocování prodeje (Business Intelligence), vzniká nutnost definovat v systému určité volitelné parametry či atributy, které se pak k produktům spolu s jejich specifickou hodnotou přiřadí. Pomocí parametrů lze např. u textilního sortimentu napříč všemi produkty uvádět barvy, velikosti, použité materiály apod. v jednotném formátu. Jako hodnoty mohou být použity různé datové typy jako číslo, datum, řetězec, ale i výčtový typ. Opět je třeba brát zřetel na lokalizaci, a tak textové hodnoty musí být možno evidovat v různých jazycích. Představme si tedy parametry jako jednotlivé dimenze ve znalostním prostoru k produktu.

Vedle takto unifikovaných a vesměs funkčně směrovaných definicí hodnot parametrů lze díky současným technologiím příslušné parametry namapovat na parametry v ontologii jiného subjektu, a tak propojit významově stejné parametry pro automatické vyhodnocování a párování např. při importu produktů. Spíše než dohledávání shodných parametrů v ontologiích mezi jednotlivými subjekty se dnes využívá veřejných ontologií, které již obsahují definice pro zařazení (taxonomii) a použití příslušných parametrů k desítkám až stovkám tisíc různých druhů zboží. Je pak rozhodně jednodušší a dlouhodobě udržitelnější propojit svoje lokální třídy či entity (jedince) s nějakou veřejně uznávanou a nejlépe standardizovanou ontologii (v pozici prostředníka), kde je větší pravděpodobnost, že takové propojení budou realizovat i ostatní účastníci řetězce.

**Varianty**

Často je podmínkou pro prodej určitého typu zboží vybrat např. velikost (u triček, bot atp.), obecně však jakoukoliv hodnotu či několik hodnot parametrů, které jednoznačně od sebe odlišují různé modely (varianty) v rámci jednoho druhu produktu. Pokud budeme uvažovat parametry jako již avizované dimenze, potom varianta je průsečík v prostoru reprezentující hodnoty od všech použitých dimenzí. Pro účely optimalizace systém eviduje varianty pouze jako množiny parametrů, jejichž hodnoty se pro jednotlivé parametry liší. Výběr určité varianty pro nákup může být reprezentován způsobem na *Obr.1: Grafický výběr varianty* Grafický výběr varianty.




**Obr.1: Grafický výběr varianty**

Protože každá varianta reprezentuje konkrétní zvolený model zboží, přichází potřeba uchovávat informace o skladových zásobách a cenách

pro jednotlivé varianty (v kombinaci s konkrétním skladem a zákaznickou skupinou – bude diskutováno později), proto také nejsou např. ceny spojeny přímo s obecným produktem. Tento fakt ve výsledku způsobuje, že jednotlivé objednávky se vztahují k variantám namísto produktů. Výše zmíněné definice varianty navíc implikují dvě logické podmínky: varianta se může vztahovat vždy k právě jednomu produktu a produkt musí obsahovat alespoň jednu (generickou) variantu.

Většina současných skladových systémů však umožňuje pouze vytvoření oddělených samostatných produktů a následné propojení relací informující, že tyto záznamy symbolizují jednotlivé varianty od určitého produktu. Správnost takového řešení je diskutabilní a z hlediska logického uspořádání je nevhodné. Samozřejmě může být výhodnější zboží s menším počtem odlišných hodnot variant duplikovat, ale v nejhorším případě tak vytvoříme tolik jednotlivých produktů, kolik je hodnota kartézského součinu všech hodnot variantních parametrů. Ve výsledku tak systémy bez parametrických variant pro tyto účely nedostačují.

**Nabídky**

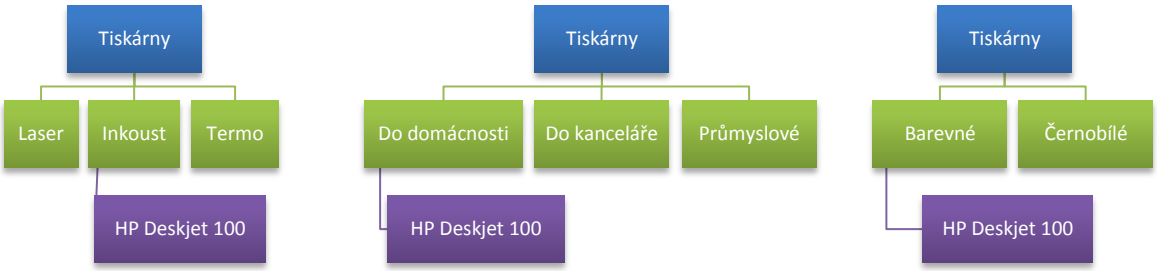
V reálném světě se běžně vyskytují případy provozu vícero skladů či prodejen v rámci společnosti, a vzniká tak požadavek na evidenci skladového hospodářství pro tyto objekty zvlášť. Správně navržený systém, který je určen pro B2B online obchod, by měl obsahovat integrovanou nabídku celého sortimentu na jednom místě a fyzické stavy či odlišnosti v dostupnosti určitého zboží pro některá místa by měly být evidovány na jiné úrovni. V navržené ontologii se proto setkáme s tzv. nabídkou (variant offer), která je průsečíkem varianty, skladu a zákaznické skupiny. To dává provozovateli široké možnosti v oblasti stanovení cen pro zákazníky s větším obratem, ale i omezení prodeje určitých variant (produktů) ve spojení s konkrétními sklady či prodejnami, příp. cenových odlišností pro prodejny v jiných městech.

Pro každou nabídku se eviduje cena s možností specifikace v různých měnách, dále počet jednotek skladem a informace, na kterém místě se nachází. Pro sledování historie cen zde figuruje i platnost nabídky, takže je možné ukládat změny pro určitý druh nabídky se zachováním starších hodnot pro pozdější prodejní optimalizace či kontrolu.

**Zařazení**

Asi nejdůležitější informací pro popis produktů je jejich zařazení do stromové struktury kategorií, označované také jako taxonomie. Ideální, resp. jednotná cesta ve stromě pro nalezení produktu neexistuje – ke každé položce lze s ohledem na její vlastnosti obecně dojít různými způsoby. *Obr.2: Různé možnosti kategorizace produktů* znázorňuje tuto

nejednoznačnost. Obyčejně je nabízeno několik cest, pomocí kterých je uživatel navigován (dle druhu, dle výrobce, dle cenového rozpětí) a zařazení do takové struktury úzce souvisí s parametry produktů, jelikož s jejich pomocí lze automatizovaně vyhledávat další skupiny a způsoby třídění. Podobně jako u parametrů může být výhodné mapování jednotlivých kategorií (ve smyslu entit v databázi) na entity nějaké veřejně uznávané ontologie (např. eCl@ss, GoodRelations aj.). Spojením kategorií s obsaženými produkty a jejich parametry lze dynamicky tvořit i filtry, které napomohou k rychlejší orientaci a nalezení konkrétního produktu.




**Obr.2: Různé možnosti kategorizace produktů**

Jak již bylo uvedeno, stromů může být několik, ale nemusí sestávat vždy nutně z jednoho typu kategorií. Pro každou kategorii ve stromě lze přiřadit nějaký funkční typ, který ji odlišuje od jejích bezprostředních sousedů. V mnoha případech z praxe je totiž žádoucí, aby se tyto speciální kategorie zobrazovaly jiným způsobem, popř. na jiném místě (výprodej, novinky, akce…). Mnou navržený systém tak umožňuje přiřadit kategoriím různé uživatelsky definované typy, s kterými pak může nejen uživatelské rozhraní dále pracovat.

Další vlastností mého modelu je samozřejmě možnost přiřadit jeden produkt do více kategorií zároveň, stejně tak jedna kategorie může obsahovat více produktů najednou (M:N kardinalita relace).

Skupina, resp. kategorie má podobně jako produkty některé povinné parametry, a to:

- Jednoznačný identifikátor (ID)
- Název (v různých jazycích)
- Textový popis (v různých jazycích)
- Priorita (pořadí v rámci nadskupiny)
- Typ (definovaný v systému pro znovupoužití)

**Obrázky a odkazy**

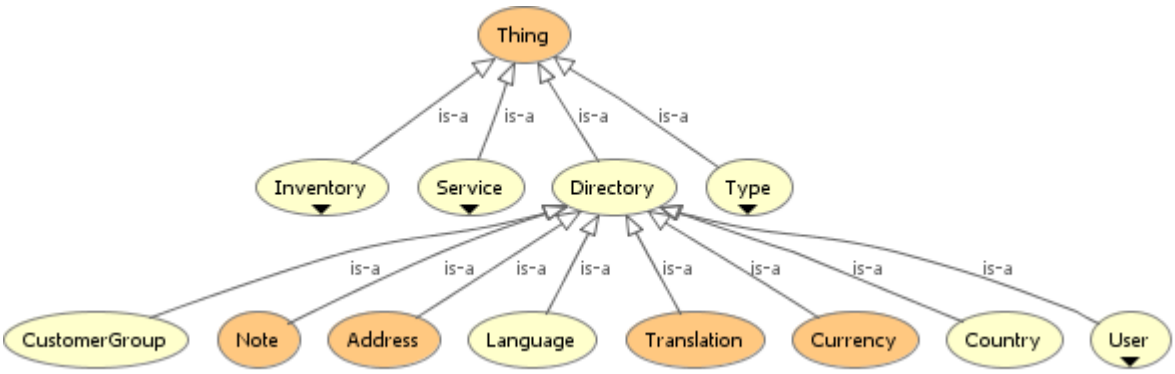
Posledními prvky, které obohacují informační povědomí o produktech, jsou odkazy na další webové zdroje. Ke každému produktu musí být možnost definovat množinu odkazů na zdroje opět v závislosti na použitém jazyku. Pro odlišení různých typů odkazů (obrázky, návody, brožury, ovladače a další) slouží obdobně jako u kategorií uživatelsky definovaný typ. Další podobností s kategoriemi je funkce pro řazení těchto zdrojů v rámci svých skupin – pomocí priority lze např. odvodit, který obrázek z galerie je pro určitý produkt hlavní.

# 6 Realizace řešení

## 6.1 Ontologie online B2B služeb

Pro tvorbu ontologie jsem si vybral open-source nástroj Protégé (konkrétně ve verzi 4.1.0), který poskytuje kompletní vizuální prostředí vč. interaktivních schémat, grafů, nástrojů pro odvozování souvislostí a kontrolu chyb (reasonerů) či dalších pluginů. S ohledem na navržené služby a mnou definované požadavky na znalostní reprezentaci produktů jsem rozdělil vytvořené třídy do následující logické struktury:

- **Directory** (= adresář, obsahuje adresy, jazyky, měny, uživatele…)
- **Inventory** (podtřídy zajišťují evidenci sortimentu)
- **Service** (modely a podpůrné třídy pro poskytované služby)
- **Type** (uživatelské typy pro různé objekty)




**Obr.3: Schéma nejvyšších tříd v ontologii**

*Obr.3: Schéma nejvyšších tříd v ontologii* ukazuje návrh struktury ontologie do 4 hlavních tříd vč. bezprostředních potomků třídy Directory pro ilustraci. Vedle jednotlivých tříd jsou součástí ontologie samozřejmě i vztahy mezi objekty a atributy s hodnotami specifikovanými příslušnými datovými typy. Vztahy a atributy obsahují ve většině případů omezení pro použití v rámci určených tříd nebo další doplňující atributy – u vzájemných vztahů (např. typu potomek-rodič) jsem tyto označil jako inverzní, a tedy při zkoumání ontologie budou jednoznačně definovány v obousměrném vztahu. Určitá skupina vazeb získala tzv. funkční atribut, kde každá třída, která chce tuto relaci využít, musí implementovat právě jednu takovou vazbu (např. varianta musí být potomkem právě jednoho produktu, proto je vztah isVariantOf označen jako funkční).

**Tab.1: Nejdůležitější třídy pro evidenci produktů**

| Zařazení třídy v ontologii | Popis a funkční přiřazení |
|---|---|
| Inventory > Product | Reprezentuje konkrétní produkt |
| Inventory > Parameter | Parametr k produktu – označen názvem, typem a měrnou jednotkou |
| Inventory > Variant | Varianta – množina hodnot parametrů přiřazených k produktům; může mít definovánu doporučenou nebo běžnou cenu, hmotnost či měrné množství |
| Inventory > ProductGroup | Skupina produktů, neboli kategorie |
| Inventory > ProductFile | Odkaz na soubor přiřazený k produktu |

**Tab.2: Nejdůležitější třídy reprezentující objednávky a jejich podpůrné prvky**

| Zařazení třídy v ontologii | Popis a funkční přiřazení |
|---|---|
| Directory > User | Uživatel systému (administrátor i zákazník) |
| Service > Document | Dokument vytvořený k objednávce – faktura, dobropis, dodejka… |
| Service > Offer | Nabídka varianty pro jednotlivé zákaznické skupiny a jednotlivé sklady |
| Service > Order | Objednávka – model pro interní evidenci |
| Service > OrderItem | Položka (zakoupená varianta, resp. vybraná nabídka) v objednávce |
| Service > OrderStage | Stav objednávky - obsahuje typ, uživatele, který jej vytvořil, datum přechodu… |
| Service > Store | Reprezentace skladu, prodejny atp. |

**Tab.3: Třídy reprezentující model poskytovaných online služeb**

| Zařazení třídy v ontologii | Popis a funkční přiřazení |
|---|---|
| Service > Model > OrderStatus | Dotaz na stav objednávky, přijímá ID objednávky, vrací systémový stav |
| Service > Model > ProductCatalogExport | Export katalogu produktů, vrací produkty vč. jejich parametrů, zařazení do skupin a parametrů |
| Service > Model > PurchaseOrder | Služba pro vložení objednávky do systému |
| Service > Model > StockListExport | Export skladových zásob a cen pro příslušného uživatele |

## 6.2 Tvorba databáze

Součástí práce je i návrh databáze sestavený na základě znalostí, resp. požadavků definovaných v ontologii. Postup je v souladu s běžným návrhem systému, kde v první řadě dochází ke specifikaci požadavků zadavatele, následkem kterých se vytvoří příslušná dokumentace. Ontologie díky zachycení statických i dynamických prvků systému může do jisté míry nahradit některé diagramy (usecase diagram, activity diagram aj.) a pro různé způsoby použití může být rozdělena do několika vrstev. Formálně správně sestavená ontologie tedy může dostatečně posloužit i k tvorbě databáze; některá místa však mohou být dezinterpretována a záleží tak na pohledu designéra, jakým způsobem znalostní model převede do modelu relačního, s nímž dnešní nejpoužívanější databáze pracují. Existují nástroje

i pro automatické vygenerování databáze přímo ze souboru ontologie, ale snahou bylo sestavit databázi ručně pro lepší pochopení souvislostí a rozdílů mezi oběma celky.

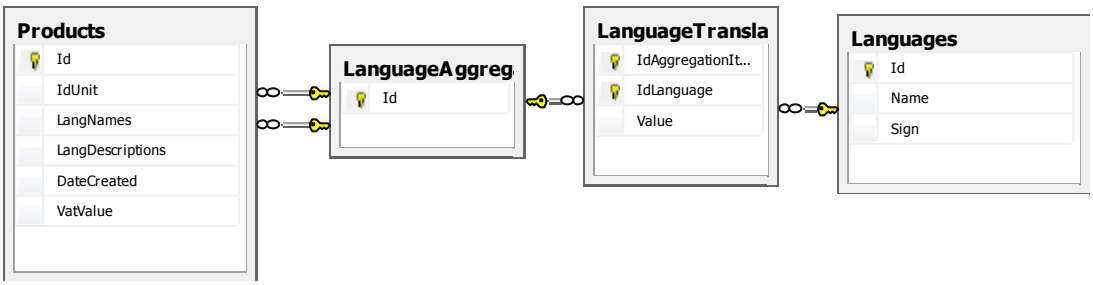
Při zkoumání ontologie, resp. procesů v ní zachycených se bylo nutné rozhodnout, zda se databáze bude ubírat směrem k typu **OLTP**[14] (transakční typ, kde dochází k častým změnám a výpočtům) či **OLAP**[15] (analytický typ obsahující historická a neměnná data), každopádně je obtížné a v některých případech i nemožné to jednoznačně určit. První ze způsobů se jeví vhodnějším, ale je potřeba vzít v úvahu i evidenci dat zachycujících různé stavy, které jsou proměnné v čase. Problém se týká zejména objednávek, kde je žádoucí s příslušným záznamem u objednaných položek zachovat název produktu (varianty), jeho cenu a další atributy aktuální v době vytvoření. Vedle problémů se zachováním původních dat čelíme horšímu problému: existence samotné entity. Kupříkladu bude v roce 2009 vytvořena objednávka na produkt A, ten se ale v roce 2012 vyřadí z nabídky a teoreticky tím dojde k jeho odstranění z tabulky *Products*. Pro zachování referenční integrity bude nutno tedy odstranit i záznam z objednaných položek, ale protože již byly některé objednávky vyřízeny a pravděpodobně došlo k fyzickému odeslání, není takové chování přípustné s ohledem na možné potřeby do budoucna. Mezi ně můžeme jmenovat třeba reklamační řízení, dohledání pohybů skladu a účetnictví nebo v dnešní době stále důležitější analytické zpracování (Business Intelligence), kdy potřeba vedení záznamů i pro již neprodávané produkty může trvat několik let (ideálně pak po celou dobu existence dat). V praxi se využívá dvou přístupů: buď se produkt fyzicky neodstraní z původního úložiště, ale nastaví se mu nějaký archivní atribut, který bude řídit např. zobrazování produktů, čímž se zachová původní datová normalizace a relační princip, nebo se ve chvíli vytvoření objednávky zkopírují potřebná data nutná k evidenci objednávek a produkt se odstraní s výše uvedenými důsledky. První z metod je náročnější na kapacitu databáze a referenční integrita může manipulaci s daty brzdit, druhý však neumožňuje např. vyhledat všechny objednávky k danému zboží, pokud již bylo odstraněno (resp. zbude možnost porovnání pouze dle názvu zboží, ale již nikoliv dle parametrů, zařazení apod.). Větší aplikace problém řeší více různými databázemi, kde provozní DB zpravidla bývá typu OLTP a data o změnách stavů či pohybech se periodicky exportují do datového skladu typu OLAP, kde se nevyužívá normálních forem a datových dekompozic. Pro tuto práci jsem zvolil hybridní způsob, kdy sice dojde k překopírování časově nebo stavově závislých dat do patřičných tabulek, ale navíc se vytvoří i relace mezi položkou objednávky a objednaným produktem –

[14] OLTP – Online Transaction Processing (online transakční zpracování)

[15] OLAP – Online Analytical Processing (online analytické zpracování)

ta zůstane platná do doby odstranění produktu, kdy se kaskádově nastaví na NULL, což nezpůsobí odstranění záznamu z objednávky a po dobu existence produktu bude možno provádět i dotazy analytického typu.

Obsluha časově modifikovaných dat však nebyla jediným problémem při tvorbě databáze. Podobně složitým z hlediska návrhu byl požadavek na lokalizaci názvů (obecně většiny textových hodnot) a měn. Musíme brát v potaz existenci libovolného počtu cizích jazyků i možnost postupného přidávání dalších či odstraňování současných překladů, což znemožňuje situaci řešit na úrovni jednotlivých sloupců v příslušných tabulkách, kde by člověk intuitivně evidoval přeložené hodnoty. Překlady i ceny tedy musí být uloženy na jiném místě, kde budou referenčně závislé na konkrétním jazyku či měně. Za tohoto předpokladu lze jednoduše vytvořit tabulka s překlady, ovšem pouze pokud její hodnoty budou sloužit k jednomu druhu překladu (např. názvu produktu). V našem případě se ale v databázi vyskytuje více hodnot určených k překladu, a to nejen v rámci jedné tabulky, ale i napříč celou databází. Abychom nemuseli situaci řešit tvorbou desítek různých překladových tabulek, zavedeme do schématu ještě jednu tabulku, která bude formou primárního klíče od sebe odlišovat jednotlivé typy překladů pro různé objekty (název produktu X, popisek kategorie Y…). Ve výsledku




**Obr.4: Znázornění způsobu reprezentace hodnot přeložitelných do ciz. jazyka**

dostaneme elegantní řešení sestavené z dvou přidaných tabulek – silně entitní agregační (typové) a překladové, kde každý záznam bude odkazovat na použitý typ překladu, konkrétní jazyk a konečně cílovou textovou hodnotu. V tabulkách, kde potřebujeme nějakou hodnotu vést v různých jazycích potom jednoduše odkážeme na typ překladu do agregační tabulky. Během dotazování na hodnotu z tohoto slovníku uvedeme požadovaný jazyk a typ překladu. Velkou výhodou řešení je evidence všech textových hodnot v jedné tabulce, nad kterou lze např. nadefinovat full textový index a dobře poslouží i pro případný export pro překladatele.

Prozatím jsem popisoval problém s překlady a tvorbou vhodné struktury pro slovník; stejný postup můžeme ale použít i pro již zmíněnou evidenci cen v různých měnách. Jediným rozdílem bude použitý datový typ pro konkrétní hodnoty cen a atribut udávající povolení

o automatickém přepočtu v případě změny měnového kurzu. Někdy je totiž žádoucí mít ceny v různých měnách pod kontrolou a spravovat je manuálně.

Schematická jednoduchost tohoto řešení je vyvážena složitější manipulací s daty. V případě tvorby nového produktu vznikne potřeba vytvořit nejprve dva typy překladů v agregační tabulce – pro název produktu a jeho popis. Následně klíče získané automatickým vygenerováním (auto increment) uložíme v záznamu s produktem. V poslední fázi transakce vložíme hodnoty do tabulky s překlady, kde ještě odkážeme na použitý jazyk a typ překladu z agregační tabulky. Díky tomu nám vznikne relace mezi produktem a hodnotou překladu, což byl záměr této metody. Bohužel se ale mezi produktem a překladem nejedná o relaci s kardinalitou umožňující kaskádové mazání nebo úpravy, proto vzniklou situaci musíme vyřešit použitím triggerů. Obecně jsou triggery využívány pro komplexnější činnosti a jejich použití ve formě udržování referenční integrity by se mělo omezit, jelikož vnáší do databázového schématu obtíže ve smyslu snadného přehlédnutí při ladění nebo pozdějších úpravách struktury zapojených tabulek. Bez triggerů by v případě odstranění záznamu např. s produktem zůstal typ překladu v agregační tabulce, kam se z produktu pouze odkazujeme cizím klíčem. Výhodou je ale, že v rámci celé databáze jsou všechny typy překladů unikátní a pro tabulky využívající překlady, resp. hodnoty měn lze vytvořit delete triggery, které budou mít za následek odstranění záznamů z agregační tabulky a kaskáda následně zajistí odstranění všech hodnot překladů, resp. měn. Naštěstí v této aplikaci nebude tak často k mazání docházet, proto je řešení vhodné.

Vedle návrhu struktury tabulek, relací a triggerů byl zásadní i výběr databázového systému (RDBMS[16]). Mezi nejpoužívanější se řadí **MySQL**, **Postgres**, **MS SQL Server** a **Oracle**. První dvě varianty jsou rozšířené především na UNIX platformách, jelikož jsou open-source a ve většině případů je jejich použití zdarma – typické použití je pro webhosting a menší aplikace. Zato Microsoft SQL Server se běžně využívá ve firemním prostředí pro střední až velké databáze díky rozmanité nabídce edicí – počínaje Express verzí, která je zadarmo až po Enterprise pro datacentra. Oracle najdeme spíše u větších systémů, a pokud se nepohybujeme v ryze korporátním prostředí, tak s ním pravděpodobně nepřijdeme do styku. Pro tuto práci jsem si vybral řešení od Microsoftu, a to díky osobním zkušenostem a tedy nejlepší znalosti oproti ostatním systémům. Zároveň se prostřednictvím .NET platformy nabízí široké možnosti integrace s vývojovým prostředím Visual Studio i aplikací samotnou.

[16] RDBMS – Relational Database Management Systém (systém pro správu relační báze dat)

Posledním úkolem při tvorbě databáze bylo její naplnění daty pro účely testování rozhraní, ale i následnou výkonovou optimalizaci. Generování takových dat umožňuje mnoho nástrojů, mezi nejznámější bych mohl jmenovat SQL Data Generator od společnosti Redgate či TurboData od Turbo Computer Systems. Bohužel většina utilit se zaměřuje spíše na generování jmen, adres apod. a žádná z mnou nalezených nedisponuje dostatečně velkým slovníkem pro generování názvů produktů. Aby mohly být výsledky optimalizace dostatečně reprezentativní, bylo potřeba naplnit databázi co nejvíce záznamy, proto bylo důležité rozvrhnout správné počty záznamů v používaných tabulkách. S ohledem na praxi jsem usoudil, že počet nabízených rozličných produktů se bude pohybovat řádově okolo 10 000. Proto jsem v *Tab.4: Rozvrh náhodně vygenerovaných dat pro produkty* definoval následující požadavky a rozmístění dat:

**Tab.4: Rozvrh náhodně vygenerovaných dat pro produkty**

| Typ | Počet záznamů |
|---|---|
| Produkty | 10 000 |
| Kategorie | 20 |
| Parametry | 20 |
| Přiřazených kategorií pro 1 produkt | 3 |
| Přiřazených parametrů pro 1 produkt | 4 |
| Počet hodnot ke každému produktovému parametru | 2 |
| Počet souborů pro 1 produkt | 1 až 5 |
| Počet variant pro 1 produkt | 6 až 8 (dle hodnot přísl. parametrů) |
| Počat nabídek pro 1 variantu | 3 (evidujeme 3 zákaznické skupiny) |
| Celkový počet nabídek | 10 000 * (6 až 8) * 3 = 180 000 až 240 000 |

Z hlediska počtů záznamů se tak stále jedná o malou databázi, kde se pohybujeme v řádech max. statisíců – toto číslo bude růst v závislosti na budoucí četnosti úprav nabídek (tj. cen, platností pro různé sklady apod.). Každopádně hodnotím rozvrh jako odpovídající ve srovnání s provozními databázemi z praxe.

Nicméně takto specifickou strukturu dat by bylo náročné vytvořit pomocí uvedených utilit a, jak jsem již zmínil, žádná z nich nenabízela slovník jmen s 10 000 položkami, které bych mohl použít pro názvy produktů. Rozhodl jsem se tedy vytvořit vlastní skript, který databázi naplní náhodnými daty. Nevýhodou je však horší čitelnost pro člověka, jelikož každý textový popisek (ať už název produktu, jeho popisek, název kategorie či parametru apod.) je sestaven z náhodně vygenerovaného řetězce o délce 6 znaků; pro kýžené účely však toto řešení dostačuje.

Během plnění databáze jsem zavedl několik uložených procedur pro tvorbu záznamů obsahujících textové popisky či ceny. Konkrétně se jedná o **INSERT_GROUP** pro vytvoření

kategorie, **INSERT_PRODUCT** pro vytvoření produktu, **INSERT_VARIANT** pro zadání varianty produktu, **INSERT_OFFER** pro vytvoření nabídky k variantě. Všechny uvedené procedury využívají dvou pomocných procedur, a to: **INSERT_TRANSLATION**, která se zadaným textovým řetězcem a identifikátorem jazyka vytvoří záznam o překladu v agregační i překladové tabulce a **INSERT_PRICE**, která pracuje analogicky s cenovými hodnotami. Pro zajištění integrity využívají uložené procedury zapouzdření do transakcí.

## 6.3 Online rozhraní

Framework .NET, jehož volba vyplynula z výběru databázového enginu, nabízí řadu užitečných nástrojů a technologií: jmenovitě třeba objektově-relační mezivrstvy jako **LINQ**[17] či **EntityFramework**, které slouží k abstrakci práce s daty – od běžných polí až po SQL dotazy. Pro dotazování do databáze stačí definovat připojení k patřičnému serveru, které se uloží do konfiguračního souboru aplikace a automaticky se vytvoří struktura obslužných tříd, které budou reprezentovat jednotlivé tabulky v databázi. V aplikačním kódu se již s databází pracuje jako s objektovým modelem bez jakékoliv účasti SQL dotazů – ty si LINQ generuje sám na pozadí.

Úkolem online rozhraní pro přístup k databázi je možnost výstupu v různých běžně užívaných formátech (XML, CSV[18] apod). Vedle použití webových služeb by tak mělo rozhraní poskytovat i datovou výměnu prostřednictvím běžných HTTP požadavků, kdy se do URL uvedou potřebné parametry vstupu a server odpověď odešle jako souborovou přílohu ke stažení. Přesto, že export dat lze provést několika způsoby, výčet funkcí zůstává stejný, a proto jsem jednotlivé metody začlenil do logické (aplikační) vrstvy, která se stará o převod dat z databáze a připravuje je pro transformaci do konkrétních formátů. Další činnosti jako modifikace HTTP hlavičky, serializaci apod. již řeší obslužné rutiny samy.

V případě využití webových služeb je situace jednoduchá: .NET se o práci se vstupem a výstupem postará sám a s ohledem na povolené protokoly v konfiguračním souboru aplikace povolí či zakáže příslušné protokoly, což se následně projeví ve vygenerovaném WSDL souboru. Služba se nachází v souboru **Gateway.asmx** v projektu **Interface** a všechny její funkce jsou zde reprezentovány statickými metodami s atributy „[WebMethod]“. Lze si povšimnout, že v definicích (odborně nazývaných také jako signatury) jednotlivých metod jsou jako návratové typy uvedeny objekty běžně používané v aplikaci. Při volání (invokaci) webové služby

[17] LINQ – Language Integrated Query

[18] CSV – Comma Separated Values (hodnoty oddělené čárkou)

následně provede .NET runtime serializaci a výsledný objekt převede standardně na výstup typu XML, který je odeslán klientovi.

Výstup ve formátu XML lze tedy pohodlně zajistit webovými službami, kde mohou být parametry předány v URL. Zbývá tak formát CSV, který je využíván pro tabulková data. Jednotlivé záznamy a hodnoty jsou zde obyčejně odděleny řádkem, resp. středníkem (či jiným oddělovačem). Omezení, se kterým se ale CSV potýká, je nemožnost reprezentace strukturovaných dat. Pokud potřebujeme do jednoho sloupce v tabulce uložit více hodnot najednou, jedinou možností eliminující použití více CSV souborů je oddělení těchto sub-hodnot dalším oddělovačem a takto lze rekurzivně postupovat do libovolné hloubky. Pro tyto případy by se ale mělo již použít XML nebo jiný strukturovaný formát (JSON aj.). Proto v této práci bude CSV využito pouze jako ukázka pro export ceníkových dat, které mají plochý model a lze je publikovat jako tabulka. Obsluhu HTTP požadavků nalezneme v souboru **Gateway.ashx**.

Diskutabilní je otázka zabezpečení přenosu dat. Samozřejmě by měl být využit protokol HTTPS z důvodu přenosu obchodních, a tedy citlivých dat. V případě volání služeb přímo pomocí GET požadavku by se heslo uživatele nemělo přenášet v nezabezpečené formě a k volání by měl být připojen digitální podpis. Využití takového zabezpečení je ale předmětem dohody zúčastněných stran, uvolnění podepisovacího algoritmu vč. předání certifikátů atp., což již tematicky nespadá do zadání práce.

# 7 Vyhodnocení

Vyhodnocení správnosti mého řešení jsem rozdělil do dvou částí. V první srovnávám navrženou ontologii s již existující a používanou ontologií od autora zabývajícího se výzkumem sémantického webu, která je taktéž zaměřena na elektronický obchod. Podstatnou část práce tvoří i návrh databáze, a protože výsledný model musí splňovat i požadavky na výkon, popisuji v druhé části vyhodnocení výkonové analýzy a optimalizace databáze a databázového volání.

## 7.1 Srovnání s GoodRelations

Jednou z veřejně dostupných ontologií zaměřených na obchodní služby je i součást projektu „GoodRelations“ týmu Prof. Martina Heppa z německé Universität der Bundeswehr, jejíž schéma nalezneme v (2). Přestože záměry mnou navržené ontologie a ontologie prezentované Prof. Heppem jsou podobné, uvedeme si zde několik zásadních rozdílů a jejich důsledky v použitelnosti. GoodRelations je komplexní služba poskytující také slovník entit podobně jako projekt eCl@ss a tímto následkem je i ontologie pro E-Commerce propracovanější a složitější co do počtu použitých tříd a vztahů. Lépe je schopna zachytit procesy díky propracovanému adresáři obchodních kontaktů a především síti odběrných míst či prodejen, které mohou mít specifikovány i detaily jako otevírací dobu. GoodRelations řeší i reklamační řízení a obsahuje více možností pro specifikaci cen (resp. nabídek variant). K produktovým parametrům lze zadat i rozsahy kvantitativních hodnot, příp. je určit jako kvalitativní pro srovnání s jinými parametry.

V čem německé schéma nedostačuje, je omezující výčet některých atributů tříd. Například třída *gr:ProductOrService* obsahuje textovou hodnotu *gr:category* či *gr:color* (tedy služba může mít definovanou barvu, což je nelogické). V mnou navržené ontologii se na kategorie a skupiny zařazení hledí jako na samostatnou strukturu, která by samozřejmě šla odvodit automaticky s ohledem na příslušné parametry (třeba směrem od největší množiny společných parametrů až po nejmenší), ale často je nutné strom kategorií vytvořit ručně vzhledem k určité zvyklosti třídění produktů. Podobně hodnota *gr:color* by se v mé ontologii uložila do seznamu hodnot parametrů přiřazených k produktu a navíc by produkt mohl obsahovat i více barev. Varianty jsou v GoodRelations zachyceny způsobem popsaným v části 0 a nejsou tolik flexibilní jako v mém návrhu, resp. pro evidenci variant se společnými atributy vznikne v úložišti zbytečná redundance. Posledním bodem, který hodnotím jako lépe navržený v mé ontologii, jsou kódy k variantám a produktům, které souvisí s již uvedeným výčtem atributů některých tříd – ontologie týmu Prof. Heppa se omezuje na kódy typu EAN, GTIN, SKU, příp. MPN. Pokud by však nastala potřeba evidovat např. více čárových kódů EAN k jedné

variantě (což se v praxi stává) nebo potřeba do systému zařadit jiný druh kódového označení produktu, schéma GoodRelations by selhalo.

Obecně v mém návrhu ontologie je začleněna vyšší míra abstrakce pro parametry, kategorie, typy apod., což ji činí jednodušší z hlediska počtu zúčastněných tříd a vztahů. Ontologie GoodRelations však detailněji specifikuje procesy a pro svou orientaci ke konkrétním případům a výčtům konkrétních atributů může být snáze pochopitelná a propojitelná s jinými ontologiemi.

## 7.2 Výkon databáze a kvalita dotazů

Pro zkoumání kvality SQL dotazů generovaných pomocí LINQu a identifikaci případných kandidátů na databázové indexy za účelem optimalizace jsem si vytvořil jednoduchý nástroj, který zachycuje veškerou komunikaci s databázovým enginem. Podobně jako pro generování dat na trhu existuje řada nástrojů (tzv. profilerů), které fungují jako určité mezivrstvy v aplikacích a dokáží vedle sledování dotazů navíc vyhodnocovat i dobu vykonávání (execution time), datovou náročnost na přenos atp. Vzhledem k tomu, že LINQ v aplikaci zastupuje datovou vrstvu, postačilo nalézt vhodné místo, kde se budou dotazy zachytávat. Objekt reprezentující celou databázi, tzv. DataContext, nabízí vlastnost Log typu Stream (resp. TextWriter), která zapisuje právě ona databázová volání vč. předávaných parametrů. Stačilo již jednoduše tento tok přesměrovat do konzolového či souborového výstupu, kde jsem byl schopen jednotlivé SQL dotazy získat. Z hlediska vygenerované syntaxe lze usoudit, že LINQ generuje dotazy kvalitativně srovnatelné s těmi, které by psal vývojář sám (v některých případech však i lepší díky na první pohled složitějším, ale efektivnějším konstrukcím). Pouze při komplexních formulacích se zapojením několika tabulek, využitím agregačních funkcí apod., může během prvního volání dojít k určitému zdržení, kdy se musí dotaz vygenerovat a optimalizovat – řešením je předkompilace takové konstrukce a s ní může již LINQ plnohodnotně fungovat.

Jednotlivé dotazy jsem následně analyzoval pomocí Microsoft SQL Server Management Studio s využitím přehledů „**Execution Plan**“ a „**Client Statistics**“. První z nich s ohledem na strukturu dotazu graficky rozvrhne potřebné činnosti a doporučí kandidátní sloupce, nad kterými je potřeba vytvořit index pro eliminaci tzv. Table Scan procesu, kdy se pro nalezení hodnoty musí projít všechny hodnoty tabulky.

Díky správnému návrhu struktury databáze a tvorbě dotazů doporučil nástroj vytvořit pouze 2 indexy: první nad sloupcem *IdProduct* v tabulce *jProductsParameterValues* s odhadovaným zrychlením zpracování dotazu o **93,2903%** a druhý nad sloupcem *IdProduct*

v tabulce *Variants* s odhadovaným zrychlením **90,4044%**. Je tedy evidentní, že v určitých dotazech se provádí vyhledávání záznamů dle identifikátoru produktu (cizího klíče do tabulky s produkty) a vytvořením indexů se záznamy seřadí i dle tohoto klíče, což přispěje k znatelnému zvýšení výkonu.

Panel Client Statistics následně zobrazuje výsledky měření pro vykonání dotazu nad existující množinou dat – nalezneme zde např. dobu vykonání dotazu v milisekundách, počet transakcí nebo počet vyměněných paketů mezi serverem a klientem. Pro co možná nejlepší výsledky je nutné provést měření přímo na zařízení, kde databázový server běží a zajistit alespoň 3 měření pro zprůměrování případných odchylek. Jednou z nejdůležitějších hodnot je „Wait time on server replies“ v sekci „Time Statistics“, což označuje dobu čekání mezi přijetím posledního paketu od klienta a odesláním prvního paketu od serveru směrem ke klientovi, tj. dobu vykonání dotazu na serveru, kterou ovlivňuje několik faktorů – od výpočetního výkonu, přes strukturu databáze, využití indexace, cacheování až po objem zpracovávaných dat a nejlépe tak vypovídá o efektivitě řešení. Vygenerování ceníku se **77 536** záznamy trvalo serveru při 20% běžné zátěži (je používán jako webový server) celkem v průměru **244 ms**, což považuji za dobrý výsledek. Celkem při tomto dotazu bylo vygenerováno **1,7 MB** dat, bylo provedeno 6 operací typu JOIN a jedna agregační funkce.

Měření v této práci probíhalo na serveru osazeným 4 jádrovým procesorem Intel Xeon X3220 s taktem 2,4 GHz, 12 GB RAM a systémem Windows Web Server 2008 R2. Databázový server byl typu Microsoft SQL Server Web Edition (64-bit) ve verzi 10.0.2531.0.

S podstatně horším výkonem jsem se setkal v případě inicializace databáze testovacími daty, která byla představena v kapitole 6.2. Proces byl spuštěn v nočních hodinách a trval necelých **9 hodin** (konkrétně 31 622 989,7316 ms). Výsledek přisuzuji vytváření dotazů v cyklech namísto využití jedné dávky dotazů, generování jednotlivých variant a nabídek (kde se prováděly náročné dotazy typu Cross Join a kterých ve výsledku je **232 608**) a určité zdržení bylo způsobeno i využitím transakcí v uložených procedurách. Rozhodně by bylo možno optimalizovat i tuto část využitím T-SQL skriptu, který by pracoval přímo v databázovém enginu namísto použitého programu v jazyce C#, jenž navíc využíval uvedený LINQ. Díky své jednoduchosti však stávající řešení postačuje a svůj účel naplnilo. Přiložená tabulka *Tab.5: Časová náročnost vytváření produktů* značí časy, za které byly vytvořeny jednotlivé druhy objektů.

**Tab.5: Časová náročnost vytváření produktů**

| Činnost | Čas zpracování |
|---|---|
| Vytvoření 20 produktových parametrů | 556 ms |
| Vytvoření 20 kategorií | 167 ms |
| Vytvoření 1 produktu | 16 ms |
| Přiřazení produktu do 3 kategorií | 33 ms |
| Přiřazení 8 parametrizovaných hodnot k produktu | 17 ms |
| Přiřazení 5 souborů k produktu | 49 ms |
| Vyhledání hodnot pro variantu, vytvoření 1 produktové varianty a 3 nabídek k ní | V průměru 151 ms |

Výpočetně nejnáročnější činností, kterou lze v celém systému provést, je export katalogu produktů. Připomeňme si požadavky pro evidenci produktů z předešlých kapitol. Dojdeme tak snadno k závěru, že sadu těchto dat není možno reprezentovat jednou běžnou dvojrozměrnou tabulkou, resp. jedním konvenčním SQL dotazem složeným z připojených tabulek. Každému produktu může být přiřazen libovolný počet parametrů, variant, nabídek, souborů, kategorií apod. a kombinace všech těchto spojení by způsobovala násobný nárůst objemu vrácených dat a zbytečné duplicity v některých sloupcích, pokud bychom se rozhodli tabulku vytvořit dynamicky, aby obsahovala všechny zúčastněné sloupce z připojených tabulek. Proto jsem přistoupil k alternativnímu řešení, a to volání více dotazů pro každý produkt. Zásadní bylo minimalizovat počet doplňujících dotazů, ale logicky bylo možno dojít k počtu sub-dotazů stejnému, jako byl počet připojených tabulek, tj. 4 (kategorie, parametry, varianty, soubory). Tyto se musí provádět v cyklu pro každý produkt a pro celkový export do databáze zašleme (počet produktů)*4 dotazů, což je značně neefektivní a takřka nepoužitelné. Během testování tohoto přístupu jsem zjistil, že vygenerování pouhých **10 produktů** do konečné XML podoby trvá necelých **6 sekund**. Nutno ale brát v potaz, že jsem v pozici klienta využíval vzdálené připojení k databázovému serveru prostřednictvím VPN, která dle odhadu zpomaluje předávání dat asi 10 násobně oproti práci přímo na serveru, každopádně by ale vygenerování katalogu metodou vnořených dotazů trvalo desítky minut pro 10 000 produktů.

Naštěstí MS SQL Server od verze 2008 nabízí funkcionalitu označovanou jako „FOR XML PATH“, která zajišťuje stavbu XML dokumentu přímo jako výstup z databáze. Pomocí odvozených tabulek (derived tables) můžeme specifikovat strukturu takového dokumentu a ke každému použitému sloupci přiřadit informaci ve smyslu použití jako atribut či tag a jeho název. Příkaz „FOR XML“ se uvádí až na konec celého dotazu a má za následek onu transformaci z tabulky do XML struktury. Vzhledem k tomu, že tato metoda je již reprezentována jedním SQL dotazem, mohl jsem otestování provést přímo na serverové

konzoli a výsledky jsou nesrovnatelně lepší, než při volání vnořených dotazů v cyklu klienta. Panel „Client Statistics“ udává po 3 měřeních průměrnou dobu zpracování pouhých **595 ms**, přičemž bylo vygenerováno **34 MB** dat. Určitou mírou se na tomto výsledku podílejí i 2 dříve vytvořené indexy pro rychlejší vyhledání parametrů a variant k produktu. Na rozdíl od ostatních služeb jsem z tohoto dotazu vytvořil tabulkovou funkci, která je tak zapouzdřena v databázovém serveru a klientská aplikace se tak o konstrukci dotazu nemusí starat a pouze jí zašle identifikátor výstupního jazyka jako parametr. SQL kód popisovaného dotazu je uveden v příloze práce.

*Obr.5: Panel Client Statistics v SQL Management Studiu* ukazuje výsledky měření vykonání dotazu pro export katalogu a transformace do XML. S tímto výsledkem konstatuji, že je databáze optimalizována s ohledem na použití v rámci služeb zde specifikovaných.

| | Trial 3 | Trial 2 | Trial 1 | Average |
|---|---|---|---|---|
| Rows returned by SELECT statements | 10000 → | 10000 → | 10000 → | 10000.0000 |
| Number of transactions | 0 → | 0 → | 0 → | 0.0000 |
| Network Statistics | | | | |
| Number of server roundtrips | 1 → | 1 → | 1 → | 1.0000 |
| TDS packets sent from client | 1 → | 1 → | 1 → | 1.0000 |
| TDS packets received from server | 8898 → | 8898 → | 8898 → | 8898.0000 |
| Bytes sent from client | 2432 → | 2432 → | 2432 → | 2432.0000 |
| Bytes received from server | 3.64428E+... → | 3.64428E+... → | 3.64428E+... → | 36442800.0000 |
| Time Statistics | | | | |
| Client processing time | 4357 ↑ | 4324 ↓ | 4405 → | 4362.0000 |
| Total execution time | 4997 ↑ | 4941 ↑ | 4933 → | 4957.0000 |
| Wait time on server replies | 640 ↑ | 617 ↑ | 528 → | 595.0000 |

**Obr.5: Panel Client Statistics v SQL Management Studiu**

# 8 Závěr a shrnutí

V této práci jsem poukázal na některé dnešní problémy malých a středních podniků (SMB[19]) a navrhnul část řešení pro evidenci katalogu produktů a některých B2B služeb s využitím moderních přístupů, jmenovitě sémantického webu, webových služeb a technologií s nimi souvisejícími.

Během tvorby ontologie zachycující chování v B2B obchodní korespondenci jsem si uvědomil rozmanitost, následkem které mohou mít dvě firmy naprosto odlišné procesy i způsoby evidence dat a tedy i důležitost zavedení ontologií a jejich vzájemnému mapování (tzv. ontology matching). Služby navržené v této práci jsou samozřejmě jen základním výčtem toho, co lze s využitím ontologií a webových služeb pro B2B styk zajistit. Rozšíření tohoto portfolia pro plnohodnotné použití by spočívalo v doplnění webových služeb o sémantické atributy, což vyžaduje vytvoření dalších pomocných ontologií, případně doplnění stávající ontologie na úroveň kompletní definice předávaných formátů dat. Zároveň by vznikla potřeba pro definici dalších služeb, jako např. export pohybů skladu, export parametrů produktů nebo služeb propojených s dalšími službami (tzv. service composition) třeba dopravců, pomocí kterých by odběratel mohl sledovat stav zásilky.

Vedle definice služeb bylo důležité správně určit potřeby pro evidenci nabízených produktů, aby byla jejich reprezentace dostatečně abstraktní a byla schopna pojmout různé druhy zboží a jejich metadat. Tuto část tvorby považuji za nejnáročnější, jelikož bylo nutno s ohledem na praxi rozmyslet co nejvíce možných situací, které mohou důsledkem odlišnosti zboží v reálném světě nastat a takto v iteracích upravovat návrh schématu.

Navrženou ontologii jsem posléze převedl do podoby relační databáze typu MS SQL Server 2008 Web Edition. Je nutné zmínit, že se nejedná o kopii typu 1:1, už z toho důvodu, že objektové a relační vztahy či dědičnost mají v obou technologiích určitá specifika. Navíc jsem databázi doplnil o některé tabulky, které se v ontologii nevyskytují a mají sloužit spíše pro náhled toho, kam by se mohla databáze rozšiřovat, konkrétně se jedná např. interní evidenci skladového hospodářství či přechody mezi stavy objednávek, které nejsou potřeba publikovat obchodním partnerům v ontologii. Lze tedy říci, že prezentovaná ontologie bude sloužit spíše jako rozhraní poskytující představu o procesech a vztazích mezi použitými objekty v obchodním styku, nikoliv pro mapování interního know-how společnosti.

[19] SMB – Small or Medium Business (malá či střední společnost)

Pro vytvoření funkčního rozhraní jsem si zvolil jazyk C# v architektuře ASP.NET a během vývoje této webové aplikace jsem do databáze podle potřeby doplnil uložené procedury pro vkládání různých záznamů. Před výkonnostní analýzou jsem databázi naplnil náhodně vygenerovanými hodnotami čítající na 10 000 produktů a cca 230 000 položek v ceníku. Pro tento účel jsem si napsal vlastní program, který je přiložen k práci.

Posledním bodem v praktické části bylo vyhodnocení výkonu databáze a její optimalizace. Při zkoumání jednotlivých SQL dotazů vygenerovaných na pozadí použitou technologií LINQ jsem vytvořil na patřičných místech indexy pro rychlejší třídění záznamů a v případě exportu produktového katalogu pak s ohledem na výpočetní náročnost nalezl optimální řešení poskytující funkcí MS SQL Serveru pro přímý výstup dat v XML.

Možným rozšířením do budoucna, které mám zájem zpracovat, je napojení jednotlivých vyskytujících se entit a tříd k objektům poskytovaným veřejnými adresáři jako eCl@ss, GoodRelations aj., díky čemuž se podstatně zvýší možnost okamžitého napojení na objekty spolupracujících subjektů, jelikož pravděpodobně i oni budou mít své objekty napojeny na tyto adresáře. Jako úložiště mapování lze použít databázi, kde by k jednotlivým produktům, parametrům a kategoriím, které jsou řekněme těmito adresáři standardizované, přibyly odkazy na URI konkrétních entit v těchto adresářích. Obecně by se tak např. každé kategorii mohly přiřadit ekvivalenty v několika různých adresářích.

Výsledkem celé práce je tak ontologie, která může obchodním partnerům napomoci při identifikaci a integraci B2B procesů a také funkční model databáze skladu s optimalizací vč. online rozhraní pro přístup k datům prostřednictvím webových služeb a dalších komunikačních kanálů. Nad rámec zadání byla provedena již zmíněná optimalizace databáze, ale tento krok považuji za nutný pro její tvorbu.

Všechny uvedené výstupy samozřejmě zachycují nejdůležitější myšlenky s ohledem na kapacitu práce. Dle mého názoru je v tomto směru budoucnost nejen na poli B2B a chtěl bych se tak této činnosti nadále věnovat i pro své soukromé projekty, díky čemuž mohou získat technologicky náskok a lepší konkurenceschopnost na trhu. Zároveň nasazení takového konceptu může napomoci osvětě a dalšímu rozvoji technologie. Nutno totiž poznamenat, že v současnosti neexistuje jednoznačné akceptovatelné řešení ve formě standardu a každá aplikace s využitím prvků sémantického webu zvětšuje záběr a motivaci ostatních provozovatelů k jeho dalšímu rozšiřování. Je to stále však otázkou budoucnosti, nicméně již dnes lídři jako Google či Yahoo sémantický web podporují a zvýhodňují projekty, které tuto technologii implementují.

# Seznam použité literatury


1. **Fensel, Dieter.** *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce.* Berlin : Springer, 2004. 3-540-00302-9.

2. **Hepp, Martin.** The GoodRelations Ontology for E-Commerce. *GoodRelations.* [Online] 1. 10 2011. [Citace: 29. 2 2012.] http://www.heppnetz.de/ontologies/goodrelations/goodrelations-UML.png.

3. **Kunder, Maurice de.** The size of the World Wide Web (The Internet). *WorldWideWebSize.com.* [Online] 9. 1 2012. [Citace: 23. 1 2012.] http://www.worldwidewebsize.com/.

4. **SEOmoz.** Google Algorithm Change History. *seomoz.* [Online] 3. 2 2012. [Citace: 18. 2 2012.] http://www.seomoz.org/google-algorithm-change#2011.

5. **Miniwatts Marketing Group.** World Internet Usage Statistics News and World Population Stats. *Internet World Stats.* [Online] 15. 2 2012. [Citace: 19. 2 2012.] http://www.internetworldstats.com/stats.htm.

6. **Omelayenko, Borys.** Integration of Product Ontologies for B2B. *Sigecom.* [Online] [Citace: 15. 3 2012.] http://www.sigecom.org/exchanges/volume_2/2.1-Omelayenko.pdf.

7. **Hepp, Martin.** Products and Services Ontologies: A Methodology for Deriving OWL Ontologies from Industrial Categorization Standards. *Heppnetz.de.* [Online] 3 2006. [Citace: 29. 2 2012.] http://www.heppnetz.de/files/IJSWIS-eclassOWL-APA-Style-2005-final-11-17-Web.pdf.

8. **Baron, John P.** B2B Procurement. *Center for IT and e-Business Management University of Illinois at Urbana-Champaign.* [Online] 5 2000. [Citace: 10. 3 2012.] http://citebm.business.illinois.edu/mba405/guest_speaker/E-catalogs.pdf.

9. **Bussler, Christoph a Zaremba, Michal.** Usage Scenario: Semantic Web-enabled Business Protocol Standards. *The DARPA Agent Markup Language Homepage.* [Online] 8. 12 2003. [Citace: 10. 1 2012.] http://www.daml.org/services/use-cases/architecture/B2BUseCase-Zaremba.htm.

10. **Fensel, Dieter.** Product Data Integration in B2B E-Commerce. *Department of Electrical Engineering & Computer Science.* [Online] 2001. [Citace: 10. 4 2012.] http://www.cs.ucf.edu/~kienhua/classes/COP6730/E-Commerce2.pdf.

11. **Horridge, Matthew.** A Practical Guide To Building OWL Ontologies Using Protégé 4. *OWL @ Manchester.* [Online] 17. 10 2007. [Citace: 20. 12 2011.]

http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_1.pdf.

12. **Astrova, Irina.** Storing OWL Ontologies in SQL Relational Databases. *World Academy of Science.* [Online] 2007. [Citace: 4. 5 2012.] http://www.waset.org/journals/waset/v29/v29-31.pdf.

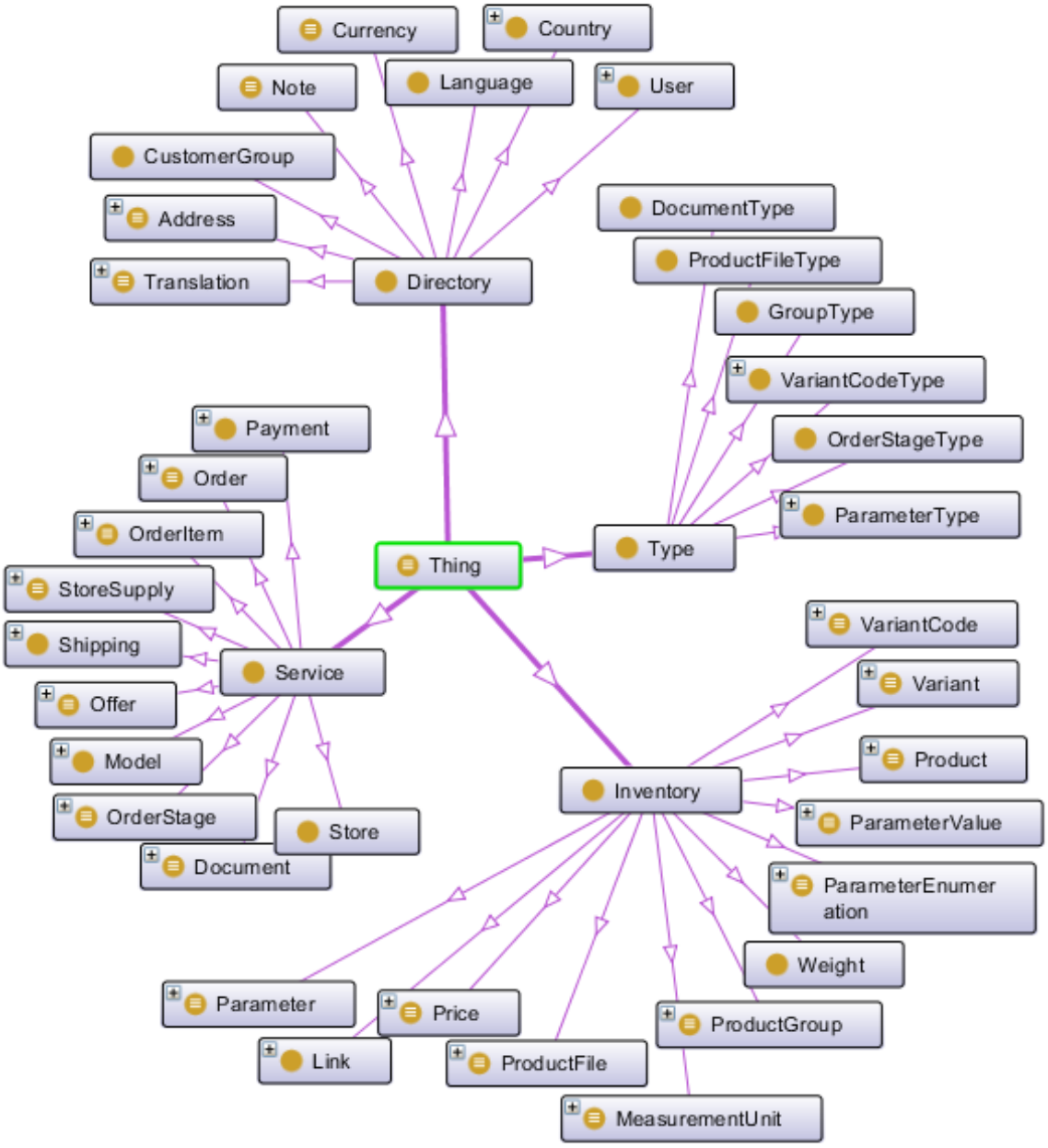
13. **Yu, Liyang.** *Introduction to The Semantic Web and Semantic Web Services.* Atlanta : Chapman & Hall/CRC, 2007. 978-15-848-8933-5.

14. **Alonso, Gustavo.** *Web Services: Concepts, Architectures and Applications.* Berlin : Springer, 2010. 978-36-420-7088-4.

15. **Fensel, Dieter.** *Enabling Semantic Web Services: The Web Service Modeling Ontology.* Berlin : Springer, 2010. 978-36-420-7088-4.

# Příloha A – Vizualizace ontologie

Schéma ukazuje 3 úrovňovou strukturu tříd se vztahy typu subClassOf obsažených v ontologii. Obrázek byl vytvořen v nástroji Protégé 4.1.0 s využitím pluginu OntoGraf 1.0.1.




**Obr. A-6: Vizualizace ontologie**

# Příloha B – SQL dotaz pro vygenerování katalogu produktů

```
SELECT
	p.Id '@Id',
	lt.value 'Name',
	lt2.value 'Description',
	(
		SELECT g.Id '@Id', lt3.Value 'text()'
		FROM jProductsGroups j, Groups g, LanguageTranslations lt3
		WHERE
			j.IdProduct = p.Id AND
			g.Id = j.IdGroup AND
			lt3.IdAggregationItem = g.LangNames
		FOR XML PATH('Category'), TYPE
	) 'Categories',
	(
		SELECT
			j.IdParameter '@IdParameter',
			j.Id '@IdCombination',
			j.Value 'text()'
		FROM jProductsParameterValues j
		WHERE j.IdProduct = p.Id
		FOR XML PATH ('Parameter'), TYPE
	) 'Parameters',
	(
		SELECT pft.[Type] '@Type', lt3.Value 'text()'
		FROM
			ProductFiles pf,
			LanguageTranslations lt3,
			ProductFileTypes pft
		WHERE
			pf.IdProduct = p.Id AND
			lt3.IdAggregationItem = pf.LangLinks AND
			pft.Id = pf.IdFileType
		FOR XML PATH ('File'), TYPE
	) 'Files',
	(
		SELECT v.[Weight] '@Weight',
			(
				SELECT j.IdParameterValue '@IdCombination'
				FROM jVariantsParameterValues j
				WHERE v.Id = j.IdVariant
				FOR XML PATH ('Parameter'), TYPE
			) 'Parameters'
		FROM Variants v
		WHERE v.IdProduct = p.Id
		FOR XML PATH ('Variant'), TYPE
	) 'Variants'
FROM Products p
JOIN LanguageTranslations lt ON p.LangNames = lt.IdAggregationItem
JOIN LanguageTranslations lt2 ON p.LangDescriptions =
lt2.IdAggregationItem
FOR XML PATH('Product')
```

# Příloha C – Ukázka exportu katalogu produktů ve formátu XML

```
<Product Id="4">
	<Name>QMUU</Name>
	<Description>ECSL</Description>
	<Categories>
		<Category Id="6">JNTM</Category>
		<Category Id="11">DHUH</Category>
		<Category Id="13">HIQF</Category>
	</Categories>
	<Parameters>
		<Parameter IdParameter="1" IdCombination="1">75</Parameter>
		<Parameter IdParameter="1" IdCombination="2">99</Parameter>
		<Parameter IdParameter="2" IdCombination="3">18</Parameter>
		<Parameter IdParameter="2" IdCombination="4">70</Parameter>
		<Parameter IdParameter="3" IdCombination="5">21</Parameter>
		<Parameter IdParameter="3" IdCombination="6">8</Parameter>
	</Parameters>
	<Variants>
		<Variant Weight="798.31">
			<Parameters>
				<Parameter IdCombination="18"/>
				<Parameter IdCombination="21"/>
				<Parameter IdCombination="75"/>
			</Parameters>
		</Variant>
		<Variant Weight="730.91">
			<Parameters>
				<Parameter IdCombination="21"/>
				<Parameter IdCombination="70"/>
				<Parameter IdCombination="75"/>
			</Parameters>
		</Variant>
	</Variants>
</Product>
```